STM32303C_EVAL BSP User Manual

Private Types

STM32303C_EVAL AUDIO

Generated on Wed May 31 2017 10:00:44 for STM32303C_EVAL BSP User Manual by doxygen 1.7.6.1

STM32303C_EVAL BSP User Manual

Private Constants

STM32303C_EVAL AUDIO

Generated on Wed May 31 2017 10:00:44 for STM32303C_EVAL BSP User Manual by doxygen 1.7.6.1

STM32303C_EVAL BSP User Manual

Private Macros

STM32303C_EVAL AUDIO

Generated on Wed May 31 2017 10:00:44 for STM32303C_EVAL BSP User Manual by doxygen 1.7.6.1

STM32303C_EVAL BSP User Manual

Exported Variables

STM32303C_EVAL AUDIO

Generated on Wed May 31 2017 10:00:44 for STM32303C_EVAL BSP User Manual by doxygen 1.7.6.1

STM32303C_EVAL BSP User Manual

Exported Macros

STM32303C_EVAL AUDIO

Generated on Wed May 31 2017 10:00:44 for STM32303C_EVAL BSP User Manual by doxygen 1.7.6.1

STM32303C_EVAL BSP User Manual

EEPROM_DrvTypeDef Struct Reference

Exported Types

```
#include <stm32303c_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 **70** of file **stm32303c_eval_eeprom.h**.

Field Documentation

uint32_t(* EEPROM_DrvTypeDef::Init)(void)

Definition at line 72 of file stm32303c_eval_eeprom.h.

Referenced by BSP_EEPROM_Init().

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

Definition at line 73 of file stm32303c_eval_eeprom.h.

Referenced by BSP_EEPROM_ReadBuffer().

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

Definition at line 74 of file stm32303c_eval_eeprom.h.

Referenced by BSP_EEPROM_WriteBuffer().

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

- stm32303c_eval_eeprom.h

STM32303C_EVAL BSP User Manual

LCD_DrawPropTypeDef Struct Reference

Exported Types

`#include <stm32303c_eval_lcd.h>`

Data Fields

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

Detailed Description

Definition at line **93** of file **stm32303c_eval_lcd.h**.

Field Documentation

uint32_t LCD_DrawPropTypeDef::BackColor

Definition at line 96 of file stm32303c_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 97 of file stm32303c_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 95 of file stm32303c_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:

- stm32303c_eval_lcd.h

STM32303C_EVAL BSP User Manual

Data Fields

Point Struct Reference

Exported Constants

`#include <stm32303c_eval_lcd.h>`

Data Fields

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

Detailed Description

Definition at line **114** of file **stm32303c_eval_lcd.h**.

Field Documentation

int16_t Point::X

Definition at line **116** of file **stm32303c_eval_lcd.h**.

Referenced by **BSP_LCD_DrawPolygon()**.

int16_t Point::Y

Definition at line **117** of file **stm32303c_eval_lcd.h**.

Referenced by **BSP_LCD_DrawPolygon()**.

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

- **stm32303c_eval_lcd.h**

STM32303C_EVAL BSP User Manual

Types Definitions

STM32303C-EVAL SD

Generated on Wed May 31 2017 10:00:44 for STM32303C_EVAL BSP User Manual by doxygen 1.7.6.1

STM32303C_EVAL BSP User Manual

SD_CSD Struct Reference

Exported Types

Card Specific Data: CSD Register. More...

```
#include <stm32303c_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 **114** of file **stm32303c_eval_sd.h**.

Field Documentation

__IO uint16_t SD_CSD::CardComdClasses

Definition at line 122 of file stm32303c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::ContentProtectAppli

Definition at line 144 of file stm32303c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::CopyFlag

Definition at line 146 of file stm32303c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::CSD_CRC

Definition at line 151 of file stm32303c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::CSDStruct

Definition at line 116 of file stm32303c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint32_t SD_CSD::DeviceSize

Definition at line 129 of file stm32303c_eval_sd.h.

Referenced by BSP_SD_GetCardInfo(), and SD_GetCSDRegister().

__IO uint8_t SD_CSD::DeviceSizeMul

Definition at line 134 of file stm32303c_eval_sd.h.

Referenced by BSP_SD_GetCardInfo(), and SD_GetCSDRegister().

__IO uint8_t SD_CSD::DSRImpl

Definition at line 127 of file stm32303c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::ECC

Definition at line 150 of file stm32303c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::EraseGrMul

Definition at line 136 of file stm32303c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::EraseGrSize

Definition at line **135** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::FileFormat

Definition at line **149** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::FileFormatGrouop

Definition at line **145** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::ManDeflECC

Definition at line **139** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::MaxBusClkFrec

Definition at line **121** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::MaxRdCurrentVDDMax

Definition at line **131** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::MaxRdCurrentVDDMin

Definition at line **130** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::MaxWrBlockLen

Definition at line **141** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::MaxWrCurrentVDDMax

Definition at line **133** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::MaxWrCurrentVDDMin

Definition at line **132** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::NSAC

Definition at line **120** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::PartBlockRead

Definition at line 124 of file stm32303c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::PermWrProtect

Definition at line 147 of file stm32303c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::RdBlockLen

Definition at line 123 of file stm32303c_eval_sd.h.

Referenced by BSP_SD_GetCardInfo(), and SD_GetCSDRegister().

__IO uint8_t SD_CSD::RdBlockMisalign

Definition at line 126 of file stm32303c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::Reserved1

Definition at line 118 of file stm32303c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::Reserved2

Definition at line **128** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::Reserved3

Definition at line **143** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::Reserved4

Definition at line **152** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::SysSpecVersion

Definition at line **117** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::TAAC

Definition at line **119** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::TempWrProtect

Definition at line **148** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::WrBlockMisalign

Definition at line **125** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::WriteBlockPaPartial

Definition at line **142** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::WrProtectGrEnable

Definition at line **138** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::WrProtectGrSize

Definition at line **137** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::WrSpeedFact

Definition at line **140** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

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

- **stm32303c_eval_sd.h**

STM32303C_EVAL BSP User Manual

SD_CID Struct Reference

Exported Types

Card Identification Data: CID Register. More...

```
#include <stm32303c_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 **158** of file **stm32303c_eval_sd.h**.

Field Documentation

__IO uint8_t SD_CID::CID_CRC

Definition at line 168 of file stm32303c_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint16_t SD_CID::ManufactDate

Definition at line 167 of file stm32303c_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint8_t SD_CID::ManufacturerID

Definition at line 160 of file stm32303c_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint16_t SD_CID::OEM_AppliID

Definition at line 161 of file stm32303c_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint32_t SD_CID::ProdName1

Definition at line 162 of file stm32303c_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint8_t SD_CID::ProdName2

Definition at line 163 of file stm32303c_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint8_t SD_CID::ProdRev

Definition at line 164 of file stm32303c_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint32_t SD_CID::ProdSN

Definition at line 165 of file stm32303c_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint8_t SD_CID::Reserved1

Definition at line 166 of file stm32303c_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint8_t SD_CID::Reserved2

Definition at line 169 of file stm32303c_eval_sd.h.

Referenced by SD_GetCIDRegister().

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

- **stm32303c_eval_sd.h**

STM32303C_EVAL BSP User Manual

SD_CardInfo Struct Reference

Exported Types

SD Card information. More...

```
#include <stm32303c_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 **175** of file **stm32303c_eval_sd.h**.

Field Documentation

uint32_t SD_CardInfo::CardBlockSize

Definition at line 180 of file stm32303c_eval_sd.h.

Referenced by BSP_SD_GetCardInfo().

uint32_t SD_CardInfo::CardCapacity

Definition at line 179 of file stm32303c_eval_sd.h.

Referenced by BSP_SD_GetCardInfo().

SD_CID SD_CardInfo::Cid

Definition at line 178 of file stm32303c_eval_sd.h.

Referenced by BSP_SD_GetCardInfo().

SD_CSD SD_CardInfo::Csd

Definition at line 177 of file stm32303c_eval_sd.h.

Referenced by BSP_SD_GetCardInfo().

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

- stm32303c_eval_sd.h

STM32303C_EVAL BSP User Manual

Exported Macro

STM32303C-EVAL SD

Generated on Wed May 31 2017 10:00:44 for STM32303C_EVAL BSP User Manual by doxygen 1.7.6.1

STM32303C_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**

- 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**

- w -

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

- x -

- X : **Point**

- y -

- Y : **Point**

STM32303C_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**

- 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**

- w -

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

- x -

- X : **Point**

- y -

- Y : **Point**

STM32303C_EVAL BSP User Manual

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

- _ -

- __STM32303C_EVAL_BSP_VERSION : **stm32303c_eval.c**
- __STM32303C_EVAL_BSP_VERSION_MAIN : **stm32303c_eval.c**
- __STM32303C_EVAL_BSP_VERSION_RC : **stm32303c_eval.c**
- __STM32303C_EVAL_BSP_VERSION_SUB1 : **stm32303c_eval.c**
- __STM32303C_EVAL_BSP_VERSION_SUB2 : **stm32303c_eval.c**

STM32303C_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 : **stm32303c_eval_lcd.c**
- AUDIO_ERROR : **stm32303c_eval_audio.h**
- AUDIO_I2C_ADDRESS : **stm32303c_eval_audio.h**
- AUDIO_IO_DeInit() : **stm32303c_eval.c**
- AUDIO_IO_Delay() : **stm32303c_eval.c**
- AUDIO_IO_Init() : **stm32303c_eval.c**
- AUDIO_IO_Read() : **stm32303c_eval.c**
- AUDIO_IO_Write() : **stm32303c_eval.c**
- AUDIO_OK : **stm32303c_eval_audio.h**
- AUDIO_OUT_IRQ_PREPRIO : **stm32303c_eval_audio.h**
- AUDIO_OUT_IRQ_SUBPRIO : **stm32303c_eval_audio.h**
- AUDIO_StatusTypeDef : **stm32303c_eval_audio.h**
- AUDIO_TIMEOUT : **stm32303c_eval_audio.h**
- AUDIODATA_SIZE : **stm32303c_eval_audio.h**

STM32303C_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 : **stm32303c_eval_lcd.c**
- BSP_AUDIO_OUT_ChangeBuffer() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_Error_CallBack() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_HalfTransfer_CallBack() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_Init() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_Pause() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_Play() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_Resume() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_SetFrequency() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_SetMute() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_SetOutputMode() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_SetVolume() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_Stop() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_TransferComplete_CallBack() : **stm32303c_eval_audio.c**
- BSP_COM_Init() : **stm32303c_eval.c**
- BSP_EEPROM_Init() : **stm32303c_eval_eeprom.c**
- BSP_EEPROM_M24LR64 : **stm32303c_eval_eeprom.h**

- BSP_EEPROM_M24M01 : **stm32303c_eval_eeprom.h**
- BSP_EEPROM_M95M01 : **stm32303c_eval_eeprom.h**
- BSP_EEPROM_ReadBuffer() : **stm32303c_eval_eeprom.c**
- BSP_EEPROM_SelectDevice() : **stm32303c_eval_eeprom.c**
- BSP_EEPROM_TIMEOUT_UserCallback() : **stm32303c_eval_eeprom.c**
- BSP_EEPROM_WriteBuffer() : **stm32303c_eval_eeprom.c**
- BSP_GetVersion() : **stm32303c_eval.c**
- BSP_JOY_GetState() : **stm32303c_eval.c**
- BSP_JOY_Init() : **stm32303c_eval.c**
- BSP_LCD_Clear() : **stm32303c_eval_lcd.c**
- BSP_LCD_ClearStringLine() : **stm32303c_eval_lcd.c**
- BSP_LCD_DisplayChar() : **stm32303c_eval_lcd.c**
- BSP_LCD_DisplayOff() : **stm32303c_eval_lcd.c**
- BSP_LCD_DisplayOn() : **stm32303c_eval_lcd.c**
- BSP_LCD_DisplayStringAt() : **stm32303c_eval_lcd.c**
- BSP_LCD_DisplayStringAtLine() : **stm32303c_eval_lcd.c**
- BSP_LCD_DrawBitmap() : **stm32303c_eval_lcd.c**
- BSP_LCD_DrawCircle() : **stm32303c_eval_lcd.c**
- BSP_LCD_DrawEllipse() : **stm32303c_eval_lcd.c**
- BSP_LCD_DrawHLine() : **stm32303c_eval_lcd.c**
- BSP_LCD_DrawLine() : **stm32303c_eval_lcd.c**
- BSP_LCD_DrawPolygon() : **stm32303c_eval_lcd.c**
- BSP_LCD_DrawRect() : **stm32303c_eval_lcd.c**
- BSP_LCD_DrawVLine() : **stm32303c_eval_lcd.c**
- BSP_LCD_FillCircle() : **stm32303c_eval_lcd.c**
- BSP_LCD_FillEllipse() : **stm32303c_eval_lcd.c**
- BSP_LCD_FillRect() : **stm32303c_eval_lcd.c**
- BSP_LCD_GetBackColor() : **stm32303c_eval_lcd.c**
- BSP_LCD_GetFont() : **stm32303c_eval_lcd.c**
- BSP_LCD_GetTextColor() : **stm32303c_eval_lcd.c**
- BSP_LCD_GetXSize() : **stm32303c_eval_lcd.c**
- BSP_LCD_GetYSize() : **stm32303c_eval_lcd.c**
- BSP_LCD_Init() : **stm32303c_eval_lcd.c**
- BSP_LCD_ReadPixel() : **stm32303c_eval_lcd.c**
- BSP_LCD_SetBackColor() : **stm32303c_eval_lcd.c** , **stm32303c_eval_lcd.h**

- BSP_LCD_SetFont() : **stm32303c_eval_lcd.c**
- BSP_LCD_SetTextColor() : **stm32303c_eval_lcd.c** , **stm32303c_eval_lcd.h**
- BSP_LED_Init() : **stm32303c_eval.c**
- BSP_LED_Off() : **stm32303c_eval.c**
- BSP_LED_On() : **stm32303c_eval.c**
- BSP_LED_Toggle() : **stm32303c_eval.c**
- BSP_PB_GetState() : **stm32303c_eval.c**
- BSP_PB_Init() : **stm32303c_eval.c**
- BSP_SD_Erase() : **stm32303c_eval_sd.h** , **stm32303c_eval_sd.c**
- BSP_SD_GetCardInfo() : **stm32303c_eval_sd.c** , **stm32303c_eval_sd.h**
- BSP_SD_GetStatus() : **stm32303c_eval_sd.c** , **stm32303c_eval_sd.h**
- BSP_SD_Init() : **stm32303c_eval_sd.h** , **stm32303c_eval_sd.c**
- BSP_SD_IsDetected() : **stm32303c_eval_sd.h** , **stm32303c_eval_sd.c**
- BSP_SD_ReadBlocks() : **stm32303c_eval_sd.h** , **stm32303c_eval_sd.c**
- BSP_SD_WriteBlocks() : **stm32303c_eval_sd.h** , **stm32303c_eval_sd.c**
- BSP_TSENSOR_Init() : **stm32303c_eval_tsensor.h** , **stm32303c_eval_tsensor.c**
- BSP_TSENSOR_ReadStatus() : **stm32303c_eval_tsensor.c** , **stm32303c_eval_tsensor.h**
- BSP_TSENSOR_ReadTemp() : **stm32303c_eval_tsensor.c** , **stm32303c_eval_tsensor.h**
- BUTTON_DOWN : **stm32303c_eval.h**
- BUTTON_IRQn : **stm32303c_eval.c**
- BUTTON_KEY : **stm32303c_eval.h**
- BUTTON_LEFT : **stm32303c_eval.h**
- BUTTON_MODE_EXTI : **stm32303c_eval.h**
- BUTTON_MODE_GPIO : **stm32303c_eval.h**
- BUTTON_PIN : **stm32303c_eval.c**
- BUTTON_PORT : **stm32303c_eval.c**
- BUTTON_RIGHT : **stm32303c_eval.h**

- BUTTON_SEL : **stm32303c_eval.h**
- Button_TypeDef : **stm32303c_eval.h**
- BUTTON_UP : **stm32303c_eval.h**
- ButtonMode_TypeDef : **stm32303c_eval.h**
- BUTTONn : **stm32303c_eval.h**
- BUTTONx_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- BUTTONx_GPIO_CLK_ENABLE : **stm32303c_eval.h**

STM32303C_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 : **stm32303c_eval_lcd.h**
- COM1 : **stm32303c_eval.h**
- COM_RX_AF : **stm32303c_eval.c**
- COM_RX_PIN : **stm32303c_eval.c**
- COM_RX_PORT : **stm32303c_eval.c**
- COM_TX_AF : **stm32303c_eval.c**
- COM_TX_PIN : **stm32303c_eval.c**
- COM_TX_PORT : **stm32303c_eval.c**
- COM_TypeDef : **stm32303c_eval.h**
- COM_USART : **stm32303c_eval.c**
- COMn : **stm32303c_eval.h**
- COMx_CLK_DISABLE : **stm32303c_eval.h**
- COMx_CLK_ENABLE : **stm32303c_eval.h**
- COMx_RX_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- COMx_RX_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- COMx_TX_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- COMx_TX_GPIO_CLK_ENABLE : **stm32303c_eval.h**

Generated on Wed May 31 2017 10:00:45 for STM32303C_EVAL BSP
User Manual by  1.7.6.1

STM32303C_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 : **stm32303c_eval_audio.h**
- DMA_MAX_SZE : **stm32303c_eval_audio.h**
- DOWN_JOY_EXTI_IRQn : **stm32303c_eval.h**
- DOWN_JOY_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- DOWN_JOY_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- DOWN_JOY_GPIO_PORT : **stm32303c_eval.h**
- DOWN_JOY_PIN : **stm32303c_eval.h**
- DrawProp : **stm32303c_eval_lcd.c**

STM32303C_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_M24LR64_A01 : **stm32303c_eval_eeprom.h**
- EEPROM_ADDRESS_M24LR64_A02 : **stm32303c_eval_eeprom.h**
- EEPROM_ADDRESS_M24M01 : **stm32303c_eval_eeprom.h**
- EEPROM_CMD_RDSR : **stm32303c_eval.c**
- EEPROM_CMD_READ : **stm32303c_eval.c**
- EEPROM_CMD_WRDI : **stm32303c_eval.c**
- EEPROM_CMD_WREN : **stm32303c_eval.c**
- EEPROM_CMD_WRITE : **stm32303c_eval.c**
- EEPROM_CMD_WRSR : **stm32303c_eval.c**
- EEPROM_CS_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- EEPROM_CS_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- EEPROM_CS_GPIO_PORT : **stm32303c_eval.h**
- EEPROM_CS_HIGH : **stm32303c_eval.h**
- EEPROM_CS_LOW : **stm32303c_eval.h**
- EEPROM_CS_PIN : **stm32303c_eval.h**
- EEPROM_FAIL : **stm32303c_eval_eeprom.h**
- EEPROM_I2C_Drv : **stm32303c_eval_eeprom.c**

- EEPROM_I2C_Init() : **stm32303c_eval_eeprom.c**
- EEPROM_I2C_IO_Init() : **stm32303c_eval.c** , **stm32303c_eval_eeprom.h**
- EEPROM_I2C_IO_IsDeviceReady() : **stm32303c_eval.c** , **stm32303c_eval_eeprom.h**
- EEPROM_I2C_IO_ReadData() : **stm32303c_eval.c** , **stm32303c_eval_eeprom.h**
- EEPROM_I2C_IO_WriteData() : **stm32303c_eval.c** , **stm32303c_eval_eeprom.h**
- EEPROM_I2C_ReadBuffer() : **stm32303c_eval_eeprom.c**
- EEPROM_I2C_WaitEepromStandbyState() : **stm32303c_eval_eeprom.c**
- EEPROM_I2C_WritePage() : **stm32303c_eval_eeprom.c**
- EEPROM_MAX_TRIALS : **stm32303c_eval_eeprom.h**
- EEPROM_OK : **stm32303c_eval_eeprom.h**
- EEPROM_PAGESIZE_M24LR64 : **stm32303c_eval_eeprom.h**
- EEPROM_PAGESIZE_M24M01 : **stm32303c_eval_eeprom.h**
- EEPROM_PAGESIZE_M95M01 : **stm32303c_eval_eeprom.h**
- EEPROM_SelectedDevice : **stm32303c_eval_eeprom.c**
- EEPROM_SPI_Drv : **stm32303c_eval_eeprom.c**
- EEPROM_SPI_Init() : **stm32303c_eval_eeprom.c**
- EEPROM_SPI_IO_Init() : **stm32303c_eval.c** , **stm32303c_eval_eeprom.h**
- EEPROM_SPI_IO_ReadByte() : **stm32303c_eval.c** , **stm32303c_eval_eeprom.h**
- EEPROM_SPI_IO_ReadData() : **stm32303c_eval.c** , **stm32303c_eval_eeprom.h**
- EEPROM_SPI_IO_WaitEepromStandbyState() : **stm32303c_eval.c** , **stm32303c_eval_eeprom.h**
- EEPROM_SPI_IO_WriteByte() : **stm32303c_eval.c** , **stm32303c_eval_eeprom.h**
- EEPROM_SPI_IO_WriteData() : **stm32303c_eval.c** , **stm32303c_eval_eeprom.h**
- EEPROM_SPI_IO_WriteDummy() : **stm32303c_eval_eeprom.h**
- EEPROM_SPI_ReadBuffer() : **stm32303c_eval_eeprom.c**
- EEPROM_SPI_WaitEepromStandbyState() : **stm32303c_eval_eeprom.c**

- EEPROM_SPI_WritePage() : **stm32303c_eval_eeprom.c**
- EEPROM_TIMEOUT : **stm32303c_eval_eeprom.h**
- EEPROM_WIP_FLAG : **stm32303c_eval.c**
- EEPROMAddress : **stm32303c_eval_eeprom.c**
- EEPROMDataRead : **stm32303c_eval_eeprom.c**
- EEPROMDataWrite : **stm32303c_eval_eeprom.c**
- EEPROMPageSize : **stm32303c_eval_eeprom.c**
- EVAL_COM1 : **stm32303c_eval.h**
- EVAL_COM1_CLK_DISABLE : **stm32303c_eval.h**
- EVAL_COM1_CLK_ENABLE : **stm32303c_eval.h**
- EVAL_COM1_IRQn : **stm32303c_eval.h**
- EVAL_COM1_RX_AF : **stm32303c_eval.h**
- EVAL_COM1_RX_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- EVAL_COM1_RX_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- EVAL_COM1_RX_GPIO_PORT : **stm32303c_eval.h**
- EVAL_COM1_RX_PIN : **stm32303c_eval.h**
- EVAL_COM1_TX_AF : **stm32303c_eval.h**
- EVAL_COM1_TX_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- EVAL_COM1_TX_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- EVAL_COM1_TX_GPIO_PORT : **stm32303c_eval.h**
- EVAL_COM1_TX_PIN : **stm32303c_eval.h**
- EVAL_I2Cx : **stm32303c_eval.h**
- EVAL_I2Cx_CLK_ENABLE : **stm32303c_eval.h**
- EVAL_I2Cx_ER_IRQHandler : **stm32303c_eval.h**
- EVAL_I2Cx_ER_IRQn : **stm32303c_eval.h**
- EVAL_I2Cx_EV_IRQHandler : **stm32303c_eval.h**
- EVAL_I2Cx_EV_IRQn : **stm32303c_eval.h**
- EVAL_I2Cx_FORCE_RESET : **stm32303c_eval.h**
- EVAL_I2Cx_RELEASE_RESET : **stm32303c_eval.h**
- EVAL_I2Cx_SCL_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- EVAL_I2Cx_SCL_GPIO_PORT : **stm32303c_eval.h**
- EVAL_I2Cx_SCL_PIN : **stm32303c_eval.h**
- EVAL_I2Cx_SCL_SDA_AF : **stm32303c_eval.h**
- EVAL_I2Cx_SDA_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- EVAL_I2Cx_SDA_GPIO_PORT : **stm32303c_eval.h**
- EVAL_I2Cx_SDA_PIN : **stm32303c_eval.h**
- EVAL_I2Cx_TIMEOUT_MAX : **stm32303c_eval.h**

- EVAL_SPIx : **stm32303c_eval.h**
- EVAL_SPIx_CLK_DISABLE : **stm32303c_eval.h**
- EVAL_SPIx_CLK_ENABLE : **stm32303c_eval.h**
- EVAL_SPIx_FORCE_RESET : **stm32303c_eval.h**
- EVAL_SPIx_MISO_MOSI_AF : **stm32303c_eval.h**
- EVAL_SPIx_MISO_MOSI_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- EVAL_SPIx_MISO_MOSI_GPIO_PORT : **stm32303c_eval.h**
- EVAL_SPIx_MISO_PIN : **stm32303c_eval.h**
- EVAL_SPIx_MOSI_PIN : **stm32303c_eval.h**
- EVAL_SPIx_RELEASE_RESET : **stm32303c_eval.h**
- EVAL_SPIx_SCK_AF : **stm32303c_eval.h**
- EVAL_SPIx_SCK_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- EVAL_SPIx_SCK_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- EVAL_SPIx_SCK_GPIO_PORT : **stm32303c_eval.h**
- EVAL_SPIx_SCK_PIN : **stm32303c_eval.h**
- EVAL_SPIx_TIMEOUT_MAX : **stm32303c_eval.h**

STM32303C_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() : **stm32303c_eval_audio.c**
- HAL_I2S_TxCpltCallback() : **stm32303c_eval_audio.c**
- HAL_I2S_TxHalfCpltCallback() : **stm32303c_eval_audio.c**
- hAudioOutI2s : **stm32303c_eval_audio.c**
- heval_I2c : **stm32303c_eval.c**
- heval_Spi : **stm32303c_eval.c**

STM32303C_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() : **stm32303c_eval.c**
- I2Cx_Init() : **stm32303c_eval.c**
- I2Cx_IsDeviceReady() : **stm32303c_eval.c**
- I2Cx_MspInit() : **stm32303c_eval.c**
- I2Cx_ReadBuffer() : **stm32303c_eval.c**
- I2Cx_ReadData() : **stm32303c_eval.c**
- I2Cx_WriteBuffer() : **stm32303c_eval.c**
- I2Cx_WriteData() : **stm32303c_eval.c**
- I2cxTimeout : **stm32303c_eval.c**
- I2Sx : **stm32303c_eval_audio.h**
- I2Sx_CLK_DISABLE : **stm32303c_eval_audio.h**
- I2Sx_CLK_ENABLE : **stm32303c_eval_audio.h**
- I2Sx_DIN_PIN : **stm32303c_eval_audio.h**
- I2Sx_DMAx_CHANNEL : **stm32303c_eval_audio.h**
- I2Sx_DMAx_CLK_DISABLE : **stm32303c_eval_audio.h**
- I2Sx_DMAx_CLK_ENABLE : **stm32303c_eval_audio.h**
- I2Sx_DMAx_IRQ : **stm32303c_eval_audio.h**
- I2Sx_DMAx_MEM_DATA_SIZE : **stm32303c_eval_audio.h**
- I2Sx_DMAx_PERIPH_DATA_SIZE : **stm32303c_eval_audio.h**

- I2Sx_FORCE_RESET : **stm32303c_eval_audio.h**
- I2Sx_Init() : **stm32303c_eval_audio.c**
- I2Sx_MCK_AF : **stm32303c_eval_audio.h**
- I2Sx_MCK_GPIO_PORT : **stm32303c_eval_audio.h**
- I2Sx_MCK_PIN : **stm32303c_eval_audio.h**
- I2Sx_MCK_WS_GPIO_CLK_DISABLE : **stm32303c_eval_audio.h**
- I2Sx_MCK_WS_GPIO_CLK_ENABLE : **stm32303c_eval_audio.h**
- I2Sx_MspInit() : **stm32303c_eval_audio.c**
- I2Sx_RELEASE_RESET : **stm32303c_eval_audio.h**
- I2Sx_SCK_DIN_AF : **stm32303c_eval_audio.h**
- I2Sx_SCK_DIN_GPIO_CLK_DISABLE : **stm32303c_eval_audio.h**
- I2Sx_SCK_DIN_GPIO_CLK_ENABLE : **stm32303c_eval_audio.h**
- I2Sx_SCK_DIN_GPIO_PORT : **stm32303c_eval_audio.h**
- I2Sx_SCK_PIN : **stm32303c_eval_audio.h**
- I2Sx_WS_AF : **stm32303c_eval_audio.h**
- I2Sx_WS_GPIO_PORT : **stm32303c_eval_audio.h**
- I2Sx_WS_PIN : **stm32303c_eval_audio.h**

STM32303C_EVAL BSP User Manual

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

- j -

- JOY_DOWN : **stm32303c_eval.h**
- JOY_IRQn : **stm32303c_eval.c**
- JOY_LEFT : **stm32303c_eval.h**
- JOY_MODE_EXTI : **stm32303c_eval.h**
- JOY_MODE_GPIO : **stm32303c_eval.h**
- JOY_NONE : **stm32303c_eval.h**
- JOY_PIN : **stm32303c_eval.c**
- JOY_PORT : **stm32303c_eval.c**
- JOY_RIGHT : **stm32303c_eval.h**
- JOY_SEL : **stm32303c_eval.h**
- JOY_UP : **stm32303c_eval.h**
- JOYMode_TypeDef : **stm32303c_eval.h**
- JOYn : **stm32303c_eval.h**
- JOYState_TypeDef : **stm32303c_eval.h**
- JOYx_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- JOYx_GPIO_CLK_ENABLE : **stm32303c_eval.h**

User Manual by  1.7.6.1

STM32303C_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 : **stm32303c_eval.h**
- KEY_BUTTON_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- KEY_BUTTON_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- KEY_BUTTON_GPIO_PORT : **stm32303c_eval.h**
- KEY_BUTTON_PIN : **stm32303c_eval.h**

STM32303C_EVAL BSP User Manual

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

- l -

- LCD_COLOR_BLACK : **stm32303c_eval_lcd.h**
- LCD_COLOR_BLUE : **stm32303c_eval_lcd.h**
- LCD_COLOR_BROWN : **stm32303c_eval_lcd.h**
- LCD_COLOR_CYAN : **stm32303c_eval_lcd.h**
- LCD_COLOR_DARKBLUE : **stm32303c_eval_lcd.h**
- LCD_COLOR_DARKCYAN : **stm32303c_eval_lcd.h**
- LCD_COLOR_DARKGRAY : **stm32303c_eval_lcd.h**
- LCD_COLOR_DARKGREEN : **stm32303c_eval_lcd.h**
- LCD_COLOR_DARKMAGENTA : **stm32303c_eval_lcd.h**
- LCD_COLOR_DARKRED : **stm32303c_eval_lcd.h**
- LCD_COLOR_DARKYELLOW : **stm32303c_eval_lcd.h**
- LCD_COLOR_GRAY : **stm32303c_eval_lcd.h**
- LCD_COLOR_GREEN : **stm32303c_eval_lcd.h**
- LCD_COLOR_LIGHTBLUE : **stm32303c_eval_lcd.h**
- LCD_COLOR_LIGHTCYAN : **stm32303c_eval_lcd.h**
- LCD_COLOR_LIGHTGRAY : **stm32303c_eval_lcd.h**
- LCD_COLOR_LIGHTGREEN : **stm32303c_eval_lcd.h**
- LCD_COLOR_LIGHTMAGENTA : **stm32303c_eval_lcd.h**
- LCD_COLOR_LIGHTRED : **stm32303c_eval_lcd.h**

- LCD_COLOR_LIGHTYELLOW : **stm32303c_eval_lcd.h**
- LCD_COLOR_MAGENTA : **stm32303c_eval_lcd.h**
- LCD_COLOR_ORANGE : **stm32303c_eval_lcd.h**
- LCD_COLOR_RED : **stm32303c_eval_lcd.h**
- LCD_COLOR_WHITE : **stm32303c_eval_lcd.h**
- LCD_COLOR_YELLOW : **stm32303c_eval_lcd.h**
- LCD_CS_HIGH : **stm32303c_eval.h**
- LCD_CS_LOW : **stm32303c_eval.h**
- LCD_DEFAULT_FONT : **stm32303c_eval_lcd.h**
- LCD_Delay() : **stm32303c_eval.c**
- LCD_DrawChar() : **stm32303c_eval_lcd.c**
- LCD_DrawPixel() : **stm32303c_eval_lcd.c**
- lcd_drv : **stm32303c_eval_lcd.c**
- LCD_ERROR : **stm32303c_eval_lcd.h**
- LCD_IO_Init() : **stm32303c_eval.c**
- LCD_IO_ReadData() : **stm32303c_eval.c**
- LCD_IO_WriteMultipleData() : **stm32303c_eval.c**
- LCD_IO_WriteReg() : **stm32303c_eval.c**
- LCD_NCS_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- LCD_NCS_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- LCD_NCS_GPIO_PORT : **stm32303c_eval.h**
- LCD_NCS_PIN : **stm32303c_eval.h**
- LCD_OK : **stm32303c_eval_lcd.h**
- LCD_READ_REG : **stm32303c_eval.c**
- LCD_SetDisplayWindow() : **stm32303c_eval_lcd.c**
- LCD_TIMEOUT : **stm32303c_eval_lcd.h**
- LCD_WRITE_REG : **stm32303c_eval.c**
- LED1 : **stm32303c_eval.h**
- LED1_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- LED1_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- LED1_GPIO_PORT : **stm32303c_eval.h**
- LED1_PIN : **stm32303c_eval.h**
- LED2 : **stm32303c_eval.h**
- LED2_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- LED2_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- LED2_GPIO_PORT : **stm32303c_eval.h**
- LED2_PIN : **stm32303c_eval.h**

- LED3 : **stm32303c_eval.h**
- LED3_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- LED3_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- LED3_GPIO_PORT : **stm32303c_eval.h**
- LED3_PIN : **stm32303c_eval.h**
- LED4 : **stm32303c_eval.h**
- LED4_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- LED4_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- LED4_GPIO_PORT : **stm32303c_eval.h**
- LED4_PIN : **stm32303c_eval.h**
- LED_BLUE : **stm32303c_eval.h**
- LED_GREEN : **stm32303c_eval.h**
- LED_ORANGE : **stm32303c_eval.h**
- LED_PIN : **stm32303c_eval.c**
- LED_PORT : **stm32303c_eval.c**
- LED_RED : **stm32303c_eval.h**
- Led_TypeDef : **stm32303c_eval.h**
- LEDn : **stm32303c_eval.h**
- LEDx_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- LEDx_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- LEFT_JOY_EXTI_IRQn : **stm32303c_eval.h**
- LEFT_JOY_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- LEFT_JOY_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- LEFT_JOY_GPIO_PORT : **stm32303c_eval.h**
- LEFT_JOY_PIN : **stm32303c_eval.h**
- LEFT_MODE : **stm32303c_eval_lcd.h**
- Line_ModeTypdef : **stm32303c_eval_lcd.h**

STM32303C_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 : **stm32303c_eval_lcd.c**
- MAX_WIDTH_FONT : **stm32303c_eval_lcd.c**
- MSD_ERROR : **stm32303c_eval_sd.h**
- MSD_OK : **stm32303c_eval_sd.h**

STM32303C_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 : **stm32303c_eval_lcd.c**

STM32303C_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 : **stm32303c_eval_audio.c**
- POLY_X : **stm32303c_eval_lcd.c**
- POLY_Y : **stm32303c_eval_lcd.c**
- pPoint : **stm32303c_eval_lcd.h**

STM32303C_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 : **stm32303c_eval.c**
- RIGHT_JOY_EXTI_IRQn : **stm32303c_eval.h**
- RIGHT_JOY_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- RIGHT_JOY_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- RIGHT_JOY_GPIO_PORT : **stm32303c_eval.h**
- RIGHT_JOY_PIN : **stm32303c_eval.h**
- RIGHT_MODE : **stm32303c_eval_lcd.h**

STM32303C_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 : **stm32303c_eval_sd.h**
- SD_CMD_CLR_WRITE_PROT : **stm32303c_eval_sd.h**
- SD_CMD_ERASE : **stm32303c_eval_sd.h**
- SD_CMD_ERASE_GRP_END : **stm32303c_eval_sd.h**
- SD_CMD_ERASE_GRP_START : **stm32303c_eval_sd.h**
- SD_CMD_GO_IDLE_STATE : **stm32303c_eval_sd.h**
- SD_CMD_PROG_CSD : **stm32303c_eval_sd.h**
- SD_CMD_READ_MULT_BLOCK : **stm32303c_eval_sd.h**
- SD_CMD_READ_SINGLE_BLOCK : **stm32303c_eval_sd.h**
- SD_CMD_SD_ERASE_GRP_END : **stm32303c_eval_sd.h**
- SD_CMD_SD_ERASE_GRP_START : **stm32303c_eval_sd.h**
- SD_CMD_SEND_CID : **stm32303c_eval_sd.h**
- SD_CMD_SEND_CSD : **stm32303c_eval_sd.h**
- SD_CMD_SEND_OP_COND : **stm32303c_eval_sd.h**
- SD_CMD_SEND_STATUS : **stm32303c_eval_sd.h**
- SD_CMD_SEND_WRITE_PROT : **stm32303c_eval_sd.h**
- SD_CMD_SET_BLOCK_COUNT : **stm32303c_eval_sd.h**
- SD_CMD_SET_BLOCKLEN : **stm32303c_eval_sd.h**
- SD_CMD_SET_WRITE_PROT : **stm32303c_eval_sd.h**

- SD_CMD_STOP_TRANSMISSION : **stm32303c_eval_sd.h**
- SD_CMD_UNTAG_ERASE_GROUP : **stm32303c_eval_sd.h**
- SD_CMD_UNTAG_SECTOR : **stm32303c_eval_sd.h**
- SD_CMD_WRITE_MULT_BLOCK : **stm32303c_eval_sd.h**
- SD_CMD_WRITE_SINGLE_BLOCK : **stm32303c_eval_sd.h**
- SD_COM_CRC_ERROR : **stm32303c_eval_sd.h**
- SD_CS_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- SD_CS_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- SD_CS_GPIO_PORT : **stm32303c_eval.h**
- SD_CS_HIGH : **stm32303c_eval.h**
- SD_CS_LOW : **stm32303c_eval.h**
- SD_CS_PIN : **stm32303c_eval.h**
- SD_DATA_CRC_ERROR : **stm32303c_eval_sd.h**
- SD_DATA_OK : **stm32303c_eval_sd.h**
- SD_DATA_OTHER_ERROR : **stm32303c_eval_sd.h**
- SD_DATA_WRITE_ERROR : **stm32303c_eval_sd.h**
- SD_DETECT_EXTI_IRQn : **stm32303c_eval.h**
- SD_DETECT_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- SD_DETECT_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- SD_DETECT_GPIO_PORT : **stm32303c_eval.h**
- SD_DETECT_PIN : **stm32303c_eval.h**
- SD_DUMMY_BYTE : **stm32303c_eval.c** , **stm32303c_eval_sd.c**
- SD_ERASE_RESET : **stm32303c_eval_sd.h**
- SD_ERASE_SEQUENCE_ERROR : **stm32303c_eval_sd.h**
- SD_GetCIDRegister() : **stm32303c_eval_sd.c**
- SD_GetCSDRegister() : **stm32303c_eval_sd.c**
- SD_GetDataResponse() : **stm32303c_eval_sd.c**
- SD_GoIdleState() : **stm32303c_eval_sd.c**
- SD_ILLEGAL_COMMAND : **stm32303c_eval_sd.h**
- SD_IN_IDLE_STATE : **stm32303c_eval_sd.h**
- SD_Info : **stm32303c_eval_sd.h**
- SD_IO_Init() : **stm32303c_eval.c** , **stm32303c_eval_sd.h**
- SD_IO_ReadByte() : **stm32303c_eval_sd.h** , **stm32303c_eval.c**
- SD_IO_WaitResponse() : **stm32303c_eval.c** , **stm32303c_eval_sd.h**
- SD_IO_WriteByte() : **stm32303c_eval_sd.h** , **stm32303c_eval.c**
- SD_IO_WriteCmd() : **stm32303c_eval.c** , **stm32303c_eval_sd.h**

- SD_IO_WriteDummy() : **stm32303c_eval_sd.h** , **stm32303c_eval.c**
- SD_NO_RESPONSE_EXPECTED : **stm32303c_eval.c** , **stm32303c_eval_sd.c**
- SD_NOT_PRESENT : **stm32303c_eval_sd.h**
- SD_PARAMETER_ERROR : **stm32303c_eval_sd.h**
- SD_PRESENT : **stm32303c_eval_sd.h**
- SD_RESPONSE_FAILURE : **stm32303c_eval_sd.h**
- SD_RESPONSE_NO_ERROR : **stm32303c_eval_sd.h**
- SD_SendCmd() : **stm32303c_eval_sd.c**
- SD_START_DATA_MULTIPLE_BLOCK_READ : **stm32303c_eval_sd.h**
- SD_START_DATA_MULTIPLE_BLOCK_WRITE : **stm32303c_eval_sd.h**
- SD_START_DATA_SINGLE_BLOCK_READ : **stm32303c_eval_sd.h**
- SD_START_DATA_SINGLE_BLOCK_WRITE : **stm32303c_eval_sd.h**
- SD_STOP_DATA_MULTIPLE_BLOCK_WRITE : **stm32303c_eval_sd.h**
- SdStatus : **stm32303c_eval_sd.c**
- SEL_JOY_EXTI_IRQn : **stm32303c_eval.h**
- SEL_JOY_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- SEL_JOY_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- SEL_JOY_GPIO_PORT : **stm32303c_eval.h**
- SEL_JOY_PIN : **stm32303c_eval.h**
- SET_INDEX : **stm32303c_eval.c**
- SPIx_Error() : **stm32303c_eval.c**
- SPIx_Init() : **stm32303c_eval.c**
- SPIx_MspInit() : **stm32303c_eval.c**
- SPIx_Read() : **stm32303c_eval.c**
- SPIx_Write() : **stm32303c_eval.c**
- SpixTimeout : **stm32303c_eval.c**
- START_BYTE : **stm32303c_eval.c**

User Manual by  1.7.6.1

STM32303C_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 -

- tsensor_drv : **stm32303c_eval_tsensor.c**
- TSENSOR_ERROR : **stm32303c_eval_tsensor.h**
- TSENSOR_I2C_ADDRESS : **stm32303c_eval_tsensor.h**
- TSENSOR_IO_Init() : **stm32303c_eval.c**
- TSENSOR_IO_IsDeviceReady() : **stm32303c_eval.c**
- TSENSOR_IO_Read() : **stm32303c_eval.c**
- TSENSOR_IO_Write() : **stm32303c_eval.c**
- TSENSOR_MAX_TRIALS : **stm32303c_eval_tsensor.h**
- TSENSOR_OK : **stm32303c_eval_tsensor.h**
- TSENSOR_Status_TypDef : **stm32303c_eval_tsensor.h**

STM32303C_EVAL BSP User Manual

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

- u -

- UP_JOY_EXTI_IRQn : **stm32303c_eval.h**
- UP_JOY_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- UP_JOY_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- UP_JOY_GPIO_PORT : **stm32303c_eval.h**
- UP_JOY_PIN : **stm32303c_eval.h**

STM32303C_EVAL BSP User Manual

- a -

- AUDIO_IO_DeInit() : **stm32303c_eval.c**
- AUDIO_IO_Delay() : **stm32303c_eval.c**
- AUDIO_IO_Init() : **stm32303c_eval.c**
- AUDIO_IO_Read() : **stm32303c_eval.c**
- AUDIO_IO_Write() : **stm32303c_eval.c**

- b -

- BSP_AUDIO_OUT_ChangeBuffer() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_Error_CallBack() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_HalfTransfer_CallBack() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_Init() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_Pause() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_Play() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_Resume() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_SetFrequency() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_SetMute() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_SetOutputMode() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_SetVolume() : **stm32303c_eval_audio.c**

- BSP_AUDIO_OUT_Stop() : **stm32303c_eval_audio.c**
- BSP_AUDIO_OUT_TransferComplete_CallBack() : **stm32303c_eval_audio.c**
- BSP_COM_Init() : **stm32303c_eval.c**
- BSP_EEPROM_Init() : **stm32303c_eval_eeprom.c**
- BSP_EEPROM_ReadBuffer() : **stm32303c_eval_eeprom.c**
- BSP_EEPROM_SelectDevice() : **stm32303c_eval_eeprom.c**
- BSP_EEPROM_TIMEOUT_UserCallback() : **stm32303c_eval_eeprom.c**
- BSP_EEPROM_WriteBuffer() : **stm32303c_eval_eeprom.c**
- BSP_GetVersion() : **stm32303c_eval.c**
- BSP_JOY_GetState() : **stm32303c_eval.c**
- BSP_JOY_Init() : **stm32303c_eval.c**
- BSP_LCD_Clear() : **stm32303c_eval_lcd.c**
- BSP_LCD_ClearStringLine() : **stm32303c_eval_lcd.c**
- BSP_LCD_DisplayChar() : **stm32303c_eval_lcd.c**
- BSP_LCD_DisplayOff() : **stm32303c_eval_lcd.c**
- BSP_LCD_DisplayOn() : **stm32303c_eval_lcd.c**
- BSP_LCD_DisplayStringAt() : **stm32303c_eval_lcd.c**
- BSP_LCD_DisplayStringAtLine() : **stm32303c_eval_lcd.c**
- BSP_LCD_DrawBitmap() : **stm32303c_eval_lcd.c**
- BSP_LCD_DrawCircle() : **stm32303c_eval_lcd.c**
- BSP_LCD_DrawEllipse() : **stm32303c_eval_lcd.c**
- BSP_LCD_DrawHLine() : **stm32303c_eval_lcd.c**
- BSP_LCD_DrawLine() : **stm32303c_eval_lcd.c**
- BSP_LCD_DrawPolygon() : **stm32303c_eval_lcd.c**
- BSP_LCD_DrawRect() : **stm32303c_eval_lcd.c**
- BSP_LCD_DrawVLine() : **stm32303c_eval_lcd.c**
- BSP_LCD_FillCircle() : **stm32303c_eval_lcd.c**
- BSP_LCD_FillEllipse() : **stm32303c_eval_lcd.c**
- BSP_LCD_FillRect() : **stm32303c_eval_lcd.c**
- BSP_LCD_GetBackColor() : **stm32303c_eval_lcd.c**
- BSP_LCD_GetFont() : **stm32303c_eval_lcd.c**
- BSP_LCD_GetTextColor() : **stm32303c_eval_lcd.c**
- BSP_LCD_GetXSize() : **stm32303c_eval_lcd.c**
- BSP_LCD_GetYSize() : **stm32303c_eval_lcd.c**
- BSP_LCD_Init() : **stm32303c_eval_lcd.c**

- BSP_LCD_ReadPixel() : **stm32303c_eval_lcd.c**
- BSP_LCD_SetBackColor() : **stm32303c_eval_lcd.c** , **stm32303c_eval_lcd.h**
- BSP_LCD_SetFont() : **stm32303c_eval_lcd.c**
- BSP_LCD_SetTextColor() : **stm32303c_eval_lcd.h** , **stm32303c_eval_lcd.c**
- BSP_LED_Init() : **stm32303c_eval.c**
- BSP_LED_Off() : **stm32303c_eval.c**
- BSP_LED_On() : **stm32303c_eval.c**
- BSP_LED_Toggle() : **stm32303c_eval.c**
- BSP_PB_GetState() : **stm32303c_eval.c**
- BSP_PB_Init() : **stm32303c_eval.c**
- BSP_SD_Erase() : **stm32303c_eval_sd.c** , **stm32303c_eval_sd.h**
- BSP_SD_GetCardInfo() : **stm32303c_eval_sd.h** , **stm32303c_eval_sd.c**
- BSP_SD_GetStatus() : **stm32303c_eval_sd.h** , **stm32303c_eval_sd.c**
- BSP_SD_Init() : **stm32303c_eval_sd.h** , **stm32303c_eval_sd.c**
- BSP_SD_IsDetected() : **stm32303c_eval_sd.h** , **stm32303c_eval_sd.c**
- BSP_SD_ReadBlocks() : **stm32303c_eval_sd.c** , **stm32303c_eval_sd.h**
- BSP_SD_WriteBlocks() : **stm32303c_eval_sd.h** , **stm32303c_eval_sd.c**
- BSP_TSENSOR_Init() : **stm32303c_eval_tsensor.h** , **stm32303c_eval_tsensor.c**
- BSP_TSENSOR_ReadStatus() : **stm32303c_eval_tsensor.h** , **stm32303c_eval_tsensor.c**
- BSP_TSENSOR_ReadTemp() : **stm32303c_eval_tsensor.h** , **stm32303c_eval_tsensor.c**

- e -

- EEPROM_I2C_Init() : **stm32303c_eval_eeprom.c**
- EEPROM_I2C_IO_Init() : **stm32303c_eval.c** , **stm32303c_eval_eeprom.h**

- EEPROM_I2C_IO_IsDeviceReady() : **stm32303c_eval.c** , **stm32303c_eval_eeprom.h**
- EEPROM_I2C_IO_ReadData() : **stm32303c_eval.c** , **stm32303c_eval_eeprom.h**
- EEPROM_I2C_IO_WriteData() : **stm32303c_eval.c** , **stm32303c_eval_eeprom.h**
- EEPROM_I2C_ReadBuffer() : **stm32303c_eval_eeprom.c**
- EEPROM_I2C_WaitEepromStandbyState() : **stm32303c_eval_eeprom.c**
- EEPROM_I2C_WritePage() : **stm32303c_eval_eeprom.c**
- EEPROM_SPI_Init() : **stm32303c_eval_eeprom.c**
- EEPROM_SPI_IO_Init() : **stm32303c_eval.c** , **stm32303c_eval_eeprom.h**
- EEPROM_SPI_IO_ReadByte() : **stm32303c_eval.c** , **stm32303c_eval_eeprom.h**
- EEPROM_SPI_IO_ReadData() : **stm32303c_eval.c** , **stm32303c_eval_eeprom.h**
- EEPROM_SPI_IO_WaitEepromStandbyState() : **stm32303c_eval_eeprom.h** , **stm32303c_eval.c**
- EEPROM_SPI_IO_WriteByte() : **stm32303c_eval_eeprom.h** , **stm32303c_eval.c**
- EEPROM_SPI_IO_WriteData() : **stm32303c_eval_eeprom.h** , **stm32303c_eval.c**
- EEPROM_SPI_IO_WriteDummy() : **stm32303c_eval_eeprom.h**
- EEPROM_SPI_ReadBuffer() : **stm32303c_eval_eeprom.c**
- EEPROM_SPI_WaitEepromStandbyState() : **stm32303c_eval_eeprom.c**
- EEPROM_SPI_WritePage() : **stm32303c_eval_eeprom.c**

- h -

- HAL_I2S_ErrorCallback() : **stm32303c_eval_audio.c**
- HAL_I2S_TxCpltCallback() : **stm32303c_eval_audio.c**
- HAL_I2S_TxHalfCpltCallback() : **stm32303c_eval_audio.c**

- i -

- I2Cx_Error() : **stm32303c_eval.c**

- I2Cx_Init() : **stm32303c_eval.c**
- I2Cx_IsDeviceReady() : **stm32303c_eval.c**
- I2Cx_MspInit() : **stm32303c_eval.c**
- I2Cx_ReadBuffer() : **stm32303c_eval.c**
- I2Cx_ReadData() : **stm32303c_eval.c**
- I2Cx_WriteBuffer() : **stm32303c_eval.c**
- I2Cx_WriteData() : **stm32303c_eval.c**
- I2Sx_Init() : **stm32303c_eval_audio.c**
- I2Sx_MspInit() : **stm32303c_eval_audio.c**

- l -

- LCD_Delay() : **stm32303c_eval.c**
- LCD_DrawChar() : **stm32303c_eval_lcd.c**
- LCD_DrawPixel() : **stm32303c_eval_lcd.c**
- LCD_IO_Init() : **stm32303c_eval.c**
- LCD_IO_ReadData() : **stm32303c_eval.c**
- LCD_IO_WriteMultipleData() : **stm32303c_eval.c**
- LCD_IO_WriteReg() : **stm32303c_eval.c**
- LCD_SetDisplayWindow() : **stm32303c_eval_lcd.c**

- s -

- SD_GetCIDRegister() : **stm32303c_eval_sd.c**
- SD_GetCSDRegister() : **stm32303c_eval_sd.c**
- SD_GetDataResponse() : **stm32303c_eval_sd.c**
- SD_GoIdleState() : **stm32303c_eval_sd.c**
- SD_IO_Init() : **stm32303c_eval.c** , **stm32303c_eval_sd.h**
- SD_IO_ReadByte() : **stm32303c_eval_sd.h** , **stm32303c_eval.c**
- SD_IO_WaitResponse() : **stm32303c_eval.c** , **stm32303c_eval_sd.h**
- SD_IO_WriteByte() : **stm32303c_eval.c** , **stm32303c_eval_sd.h**
- SD_IO_WriteCmd() : **stm32303c_eval.c** , **stm32303c_eval_sd.h**
- SD_IO_WriteDummy() : **stm32303c_eval_sd.h** , **stm32303c_eval.c**
- SD_SendCmd() : **stm32303c_eval_sd.c**
- SPIx_Error() : **stm32303c_eval.c**
- SPIx_Init() : **stm32303c_eval.c**

- SPIx_MspInit() : **stm32303c_eval.c**
- SPIx_Read() : **stm32303c_eval.c**
- SPIx_Write() : **stm32303c_eval.c**

- t -

- TSENSOR_IO_Init() : **stm32303c_eval.c**
- TSENSOR_IO_IsDeviceReady() : **stm32303c_eval.c**
- TSENSOR_IO_Read() : **stm32303c_eval.c**
- TSENSOR_IO_Write() : **stm32303c_eval.c**

STM32303C_EVAL BSP User Manual

- b -

- bitmap : **stm32303c_eval_lcd.c**
- BUTTON_IRQn : **stm32303c_eval.c**
- BUTTON_PIN : **stm32303c_eval.c**
- BUTTON_PORT : **stm32303c_eval.c**

- c -

- COM_RX_AF : **stm32303c_eval.c**
- COM_RX_PIN : **stm32303c_eval.c**
- COM_RX_PORT : **stm32303c_eval.c**
- COM_TX_AF : **stm32303c_eval.c**
- COM_TX_PIN : **stm32303c_eval.c**
- COM_TX_PORT : **stm32303c_eval.c**
- COM_USART : **stm32303c_eval.c**

- d -

- DrawProp : **stm32303c_eval_lcd.c**

- e -

- EEPROM_I2C_Drv : **stm32303c_eval_eeprom.c**
- EEPROM_SelectedDevice : **stm32303c_eval_eeprom.c**
- EEPROM_SPI_Drv : **stm32303c_eval_eeprom.c**
- EEPROMAddress : **stm32303c_eval_eeprom.c**
- EEPROMDataRead : **stm32303c_eval_eeprom.c**
- EEPROMDataWrite : **stm32303c_eval_eeprom.c**
- EEPROMPageSize : **stm32303c_eval_eeprom.c**

- h -

- hAudioOutI2s : **stm32303c_eval_audio.c**
- heval_I2c : **stm32303c_eval.c**
- heval_Spi : **stm32303c_eval.c**

- i -

- I2cxTimeout : **stm32303c_eval.c**

- j -

- JOY_IRQn : **stm32303c_eval.c**
- JOY_PIN : **stm32303c_eval.c**
- JOY_PORT : **stm32303c_eval.c**

- l -

- lcd_drv : **stm32303c_eval_lcd.c**
- LED_PIN : **stm32303c_eval.c**
- LED_PORT : **stm32303c_eval.c**

- p -

- pAudioDrv : **stm32303c_eval_audio.c**

- s -

- SdStatus : **stm32303c_eval_sd.c**
- SpixTimeout : **stm32303c_eval.c**

- t -

- tsensor_drv : **stm32303c_eval_tsensor.c**

STM32303C_EVAL BSP User Manual

- pPoint : **stm32303c_eval_lcd.h**

STM32303C_EVAL BSP User Manual

- AUDIO_StatusTypeDef : **stm32303c_eval_audio.h**
- Button_TypeDef : **stm32303c_eval.h**
- ButtonMode_TypeDef : **stm32303c_eval.h**
- COM_TypeDef : **stm32303c_eval.h**
- JOYMode_TypeDef : **stm32303c_eval.h**
- JOYState_TypeDef : **stm32303c_eval.h**
- Led_TypeDef : **stm32303c_eval.h**
- Line_ModeTypdef : **stm32303c_eval_lcd.h**
- SD_Info : **stm32303c_eval_sd.h**
- TSENSOR_Status_TypDef : **stm32303c_eval_tsensor.h**

STM32303C_EVAL BSP User Manual

- a -

- AUDIO_ERROR : **stm32303c_eval_audio.h**
- AUDIO_OK : **stm32303c_eval_audio.h**
- AUDIO_TIMEOUT : **stm32303c_eval_audio.h**

- b -

- BUTTON_DOWN : **stm32303c_eval.h**
- BUTTON_KEY : **stm32303c_eval.h**
- BUTTON_LEFT : **stm32303c_eval.h**
- BUTTON_MODE_EXTI : **stm32303c_eval.h**
- BUTTON_MODE_GPIO : **stm32303c_eval.h**
- BUTTON_RIGHT : **stm32303c_eval.h**
- BUTTON_SEL : **stm32303c_eval.h**
- BUTTON_UP : **stm32303c_eval.h**

- c -

- CENTER_MODE : **stm32303c_eval_lcd.h**
- COM1 : **stm32303c_eval.h**

- j -

- JOY_DOWN : **stm32303c_eval.h**
- JOY_LEFT : **stm32303c_eval.h**
- JOY_MODE_EXTI : **stm32303c_eval.h**
- JOY_MODE_GPIO : **stm32303c_eval.h**
- JOY_NONE : **stm32303c_eval.h**
- JOY_RIGHT : **stm32303c_eval.h**
- JOY_SEL : **stm32303c_eval.h**
- JOY_UP : **stm32303c_eval.h**

- l -

- LED1 : **stm32303c_eval.h**
- LED2 : **stm32303c_eval.h**
- LED3 : **stm32303c_eval.h**
- LED4 : **stm32303c_eval.h**
- LED_BLUE : **stm32303c_eval.h**
- LED_GREEN : **stm32303c_eval.h**
- LED_ORANGE : **stm32303c_eval.h**
- LED_RED : **stm32303c_eval.h**
- LEFT_MODE : **stm32303c_eval_lcd.h**

- r -

- RIGHT_MODE : **stm32303c_eval_lcd.h**

- s -

- SD_ADDRESS_ERROR : **stm32303c_eval_sd.h**
- SD_COM_CRC_ERROR : **stm32303c_eval_sd.h**
- SD_DATA_CRC_ERROR : **stm32303c_eval_sd.h**
- SD_DATA_OK : **stm32303c_eval_sd.h**
- SD_DATA_OTHER_ERROR : **stm32303c_eval_sd.h**
- SD_DATA_WRITE_ERROR : **stm32303c_eval_sd.h**
- SD_ERASE_RESET : **stm32303c_eval_sd.h**
- SD_ERASE_SEQUENCE_ERROR : **stm32303c_eval_sd.h**
- SD_ILLEGAL_COMMAND : **stm32303c_eval_sd.h**

- SD_IN_IDLE_STATE : **stm32303c_eval_sd.h**
- SD_PARAMETER_ERROR : **stm32303c_eval_sd.h**
- SD_RESPONSE_FAILURE : **stm32303c_eval_sd.h**
- SD_RESPONSE_NO_ERROR : **stm32303c_eval_sd.h**

- t -

- TSENSOR_ERROR : **stm32303c_eval_tsensor.h**
- TSENSOR_OK : **stm32303c_eval_tsensor.h**

STM32303C_EVAL BSP User Manual

- _ -

- __STM32303C_EVAL_BSP_VERSION : **stm32303c_eval.c**
- __STM32303C_EVAL_BSP_VERSION_MAIN : **stm32303c_eval.c**
- __STM32303C_EVAL_BSP_VERSION_RC : **stm32303c_eval.c**
- __STM32303C_EVAL_BSP_VERSION_SUB1 : **stm32303c_eval.c**
- __STM32303C_EVAL_BSP_VERSION_SUB2 : **stm32303c_eval.c**

STM32303C_EVAL BSP User Manual

- a -

- ABS : **stm32303c_eval_lcd.c**
- AUDIO_I2C_ADDRESS : **stm32303c_eval_audio.h**
- AUDIO_OUT_IRQ_PREPRIO : **stm32303c_eval_audio.h**
- AUDIO_OUT_IRQ_SUBPRIO : **stm32303c_eval_audio.h**
- AUDIODATA_SIZE : **stm32303c_eval_audio.h**

STM32303C_EVAL BSP User Manual

- b -

- BSP_EEPROM_M24LR64 : **stm32303c_eval_eeprom.h**
- BSP_EEPROM_M24M01 : **stm32303c_eval_eeprom.h**
- BSP_EEPROM_M95M01 : **stm32303c_eval_eeprom.h**
- BUTTONn : **stm32303c_eval.h**
- BUTTONx_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- BUTTONx_GPIO_CLK_ENABLE : **stm32303c_eval.h**

STM32303C_EVAL BSP User Manual

- c -

- COMn : **stm32303c_eval.h**
- COMx_CLK_DISABLE : **stm32303c_eval.h**
- COMx_CLK_ENABLE : **stm32303c_eval.h**
- COMx_RX_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- COMx_RX_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- COMx_TX_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- COMx_TX_GPIO_CLK_ENABLE : **stm32303c_eval.h**

STM32303C_EVAL BSP User Manual

- d -

- DMA_MAX : **stm32303c_eval_audio.h**
- DMA_MAX_SZE : **stm32303c_eval_audio.h**
- DOWN_JOY_EXTI_IRQn : **stm32303c_eval.h**
- DOWN_JOY_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- DOWN_JOY_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- DOWN_JOY_GPIO_PORT : **stm32303c_eval.h**
- DOWN_JOY_PIN : **stm32303c_eval.h**

STM32303C_EVAL BSP User Manual

- e -

- EEPROM_ADDRESS_M24LR64_A01 : **stm32303c_eval_eeprom.h**
- EEPROM_ADDRESS_M24LR64_A02 : **stm32303c_eval_eeprom.h**
- EEPROM_ADDRESS_M24M01 : **stm32303c_eval_eeprom.h**
- EEPROM_CMD_RDSR : **stm32303c_eval.c**
- EEPROM_CMD_READ : **stm32303c_eval.c**
- EEPROM_CMD_WRDI : **stm32303c_eval.c**
- EEPROM_CMD_WREN : **stm32303c_eval.c**
- EEPROM_CMD_WRITE : **stm32303c_eval.c**
- EEPROM_CMD_WRSR : **stm32303c_eval.c**
- EEPROM_CS_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- EEPROM_CS_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- EEPROM_CS_GPIO_PORT : **stm32303c_eval.h**
- EEPROM_CS_HIGH : **stm32303c_eval.h**
- EEPROM_CS_LOW : **stm32303c_eval.h**
- EEPROM_CS_PIN : **stm32303c_eval.h**
- EEPROM_FAIL : **stm32303c_eval_eeprom.h**
- EEPROM_MAX_TRIALS : **stm32303c_eval_eeprom.h**
- EEPROM_OK : **stm32303c_eval_eeprom.h**

- EEPROM_PAGESIZE_M24LR64 : **stm32303c_eval_eeprom.h**
- EEPROM_PAGESIZE_M24M01 : **stm32303c_eval_eeprom.h**
- EEPROM_PAGESIZE_M95M01 : **stm32303c_eval_eeprom.h**
- EEPROM_TIMEOUT : **stm32303c_eval_eeprom.h**
- EEPROM_WIP_FLAG : **stm32303c_eval.c**
- EVAL_COM1 : **stm32303c_eval.h**
- EVAL_COM1_CLK_DISABLE : **stm32303c_eval.h**
- EVAL_COM1_CLK_ENABLE : **stm32303c_eval.h**
- EVAL_COM1_IRQn : **stm32303c_eval.h**
- EVAL_COM1_RX_AF : **stm32303c_eval.h**
- EVAL_COM1_RX_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- EVAL_COM1_RX_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- EVAL_COM1_RX_GPIO_PORT : **stm32303c_eval.h**
- EVAL_COM1_RX_PIN : **stm32303c_eval.h**
- EVAL_COM1_TX_AF : **stm32303c_eval.h**
- EVAL_COM1_TX_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- EVAL_COM1_TX_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- EVAL_COM1_TX_GPIO_PORT : **stm32303c_eval.h**
- EVAL_COM1_TX_PIN : **stm32303c_eval.h**
- EVAL_I2Cx : **stm32303c_eval.h**
- EVAL_I2Cx_CLK_ENABLE : **stm32303c_eval.h**
- EVAL_I2Cx_ER_IRQHandler : **stm32303c_eval.h**
- EVAL_I2Cx_ER_IRQn : **stm32303c_eval.h**
- EVAL_I2Cx_EV_IRQHandler : **stm32303c_eval.h**
- EVAL_I2Cx_EV_IRQn : **stm32303c_eval.h**
- EVAL_I2Cx_FORCE_RESET : **stm32303c_eval.h**
- EVAL_I2Cx_RELEASE_RESET : **stm32303c_eval.h**
- EVAL_I2Cx_SCL_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- EVAL_I2Cx_SCL_GPIO_PORT : **stm32303c_eval.h**
- EVAL_I2Cx_SCL_PIN : **stm32303c_eval.h**
- EVAL_I2Cx_SCL_SDA_AF : **stm32303c_eval.h**
- EVAL_I2Cx_SDA_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- EVAL_I2Cx_SDA_GPIO_PORT : **stm32303c_eval.h**
- EVAL_I2Cx_SDA_PIN : **stm32303c_eval.h**
- EVAL_I2Cx_TIMEOUT_MAX : **stm32303c_eval.h**
- EVAL_SPIx : **stm32303c_eval.h**
- EVAL_SPIx_CLK_DISABLE : **stm32303c_eval.h**

- EVAL_SPIx_CLK_ENABLE : **stm32303c_eval.h**
- EVAL_SPIx_FORCE_RESET : **stm32303c_eval.h**
- EVAL_SPIx_MISO_MOSI_AF : **stm32303c_eval.h**
- EVAL_SPIx_MISO_MOSI_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- EVAL_SPIx_MISO_MOSI_GPIO_PORT : **stm32303c_eval.h**
- EVAL_SPIx_MISO_PIN : **stm32303c_eval.h**
- EVAL_SPIx_MOSI_PIN : **stm32303c_eval.h**
- EVAL_SPIx_RELEASE_RESET : **stm32303c_eval.h**
- EVAL_SPIx_SCK_AF : **stm32303c_eval.h**
- EVAL_SPIx_SCK_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- EVAL_SPIx_SCK_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- EVAL_SPIx_SCK_GPIO_PORT : **stm32303c_eval.h**
- EVAL_SPIx_SCK_PIN : **stm32303c_eval.h**
- EVAL_SPIx_TIMEOUT_MAX : **stm32303c_eval.h**

STM32303C_EVAL BSP User Manual

- i -

- I2Sx : **stm32303c_eval_audio.h**
- I2Sx_CLK_DISABLE : **stm32303c_eval_audio.h**
- I2Sx_CLK_ENABLE : **stm32303c_eval_audio.h**
- I2Sx_DIN_PIN : **stm32303c_eval_audio.h**
- I2Sx_DMAx_CHANNEL : **stm32303c_eval_audio.h**
- I2Sx_DMAx_CLK_DISABLE : **stm32303c_eval_audio.h**
- I2Sx_DMAx_CLK_ENABLE : **stm32303c_eval_audio.h**
- I2Sx_DMAx_IRQ : **stm32303c_eval_audio.h**
- I2Sx_DMAx_MEM_DATA_SIZE : **stm32303c_eval_audio.h**
- I2Sx_DMAx_PERIPH_DATA_SIZE : **stm32303c_eval_audio.h**
- I2Sx_FORCE_RESET : **stm32303c_eval_audio.h**
- I2Sx_MCK_AF : **stm32303c_eval_audio.h**
- I2Sx_MCK_GPIO_PORT : **stm32303c_eval_audio.h**
- I2Sx_MCK_PIN : **stm32303c_eval_audio.h**
- I2Sx_MCK_WS_GPIO_CLK_DISABLE : **stm32303c_eval_audio.h**
- I2Sx_MCK_WS_GPIO_CLK_ENABLE : **stm32303c_eval_audio.h**
- I2Sx_RELEASE_RESET : **stm32303c_eval_audio.h**
- I2Sx_SCK_DIN_AF : **stm32303c_eval_audio.h**
- I2Sx_SCK_DIN_GPIO_CLK_DISABLE :

stm32303c_eval_audio.h

- I2Sx_SCK_DIN_GPIO_CLK_ENABLE : **stm32303c_eval_audio.h**
- I2Sx_SCK_DIN_GPIO_PORT : **stm32303c_eval_audio.h**
- I2Sx_SCK_PIN : **stm32303c_eval_audio.h**
- I2Sx_WS_AF : **stm32303c_eval_audio.h**
- I2Sx_WS_GPIO_PORT : **stm32303c_eval_audio.h**
- I2Sx_WS_PIN : **stm32303c_eval_audio.h**

STM32303C_EVAL BSP User Manual

- j -

- JOYn : **stm32303c_eval.h**
- JOYx_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- JOYx_GPIO_CLK_ENABLE : **stm32303c_eval.h**

STM32303C_EVAL BSP User Manual

- k -

- KEY_BUTTON_EXTI_IRQn : **stm32303c_eval.h**
- KEY_BUTTON_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- KEY_BUTTON_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- KEY_BUTTON_GPIO_PORT : **stm32303c_eval.h**
- KEY_BUTTON_PIN : **stm32303c_eval.h**

STM32303C_EVAL BSP User Manual

- l -

- LCD_COLOR_BLACK : **stm32303c_eval_lcd.h**
- LCD_COLOR_BLUE : **stm32303c_eval_lcd.h**
- LCD_COLOR_BROWN : **stm32303c_eval_lcd.h**
- LCD_COLOR_CYAN : **stm32303c_eval_lcd.h**
- LCD_COLOR_DARKBLUE : **stm32303c_eval_lcd.h**
- LCD_COLOR_DARKCYAN : **stm32303c_eval_lcd.h**
- LCD_COLOR_DARKGRAY : **stm32303c_eval_lcd.h**
- LCD_COLOR_DARKGREEN : **stm32303c_eval_lcd.h**
- LCD_COLOR_DARKMAGENTA : **stm32303c_eval_lcd.h**
- LCD_COLOR_DARKRED : **stm32303c_eval_lcd.h**
- LCD_COLOR_DARKYELLOW : **stm32303c_eval_lcd.h**
- LCD_COLOR_GRAY : **stm32303c_eval_lcd.h**
- LCD_COLOR_GREEN : **stm32303c_eval_lcd.h**
- LCD_COLOR_LIGHTBLUE : **stm32303c_eval_lcd.h**
- LCD_COLOR_LIGHTCYAN : **stm32303c_eval_lcd.h**
- LCD_COLOR_LIGHTGRAY : **stm32303c_eval_lcd.h**
- LCD_COLOR_LIGHTGREEN : **stm32303c_eval_lcd.h**
- LCD_COLOR_LIGHTMAGENTA : **stm32303c_eval_lcd.h**
- LCD_COLOR_LIGHTRED : **stm32303c_eval_lcd.h**
- LCD_COLOR_LIGHTYELLOW : **stm32303c_eval_lcd.h**

- LCD_COLOR_MAGENTA : **stm32303c_eval_lcd.h**
- LCD_COLOR_ORANGE : **stm32303c_eval_lcd.h**
- LCD_COLOR_RED : **stm32303c_eval_lcd.h**
- LCD_COLOR_WHITE : **stm32303c_eval_lcd.h**
- LCD_COLOR_YELLOW : **stm32303c_eval_lcd.h**
- LCD_CS_HIGH : **stm32303c_eval.h**
- LCD_CS_LOW : **stm32303c_eval.h**
- LCD_DEFAULT_FONT : **stm32303c_eval_lcd.h**
- LCD_ERROR : **stm32303c_eval_lcd.h**
- LCD_NCS_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- LCD_NCS_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- LCD_NCS_GPIO_PORT : **stm32303c_eval.h**
- LCD_NCS_PIN : **stm32303c_eval.h**
- LCD_OK : **stm32303c_eval_lcd.h**
- LCD_READ_REG : **stm32303c_eval.c**
- LCD_TIMEOUT : **stm32303c_eval_lcd.h**
- LCD_WRITE_REG : **stm32303c_eval.c**
- LED1_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- LED1_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- LED1_GPIO_PORT : **stm32303c_eval.h**
- LED1_PIN : **stm32303c_eval.h**
- LED2_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- LED2_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- LED2_GPIO_PORT : **stm32303c_eval.h**
- LED2_PIN : **stm32303c_eval.h**
- LED3_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- LED3_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- LED3_GPIO_PORT : **stm32303c_eval.h**
- LED3_PIN : **stm32303c_eval.h**
- LED4_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- LED4_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- LED4_GPIO_PORT : **stm32303c_eval.h**
- LED4_PIN : **stm32303c_eval.h**
- LEDn : **stm32303c_eval.h**
- LEDx_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- LEDx_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- LEFT_JOY_EXTI_IRQn : **stm32303c_eval.h**

- LEFT_JOY_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- LEFT_JOY_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- LEFT_JOY_GPIO_PORT : **stm32303c_eval.h**
- LEFT_JOY_PIN : **stm32303c_eval.h**

STM32303C_EVAL BSP User Manual

- m -

- MAX_HEIGHT_FONT : **stm32303c_eval_lcd.c**
- MAX_WIDTH_FONT : **stm32303c_eval_lcd.c**
- MSD_ERROR : **stm32303c_eval_sd.h**
- MSD_OK : **stm32303c_eval_sd.h**

STM32303C_EVAL BSP User Manual

- o -

- OFFSET_BITMAP : **stm32303c_eval_lcd.c**

STM32303C_EVAL BSP User Manual

- p -

- POLY_X : **stm32303c_eval_lcd.c**
- POLY_Y : **stm32303c_eval_lcd.c**

STM32303C_EVAL BSP User Manual

- r -

- READ_STATUS : **stm32303c_eval.c**
- RIGHT_JOY_EXTI_IRQn : **stm32303c_eval.h**
- RIGHT_JOY_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- RIGHT_JOY_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- RIGHT_JOY_GPIO_PORT : **stm32303c_eval.h**
- RIGHT_JOY_PIN : **stm32303c_eval.h**

STM32303C_EVAL BSP User Manual

- s -

- SD_CMD_CLR_WRITE_PROT : **stm32303c_eval_sd.h**
- SD_CMD_ERASE : **stm32303c_eval_sd.h**
- SD_CMD_ERASE_GRP_END : **stm32303c_eval_sd.h**
- SD_CMD_ERASE_GRP_START : **stm32303c_eval_sd.h**
- SD_CMD_GO_IDLE_STATE : **stm32303c_eval_sd.h**
- SD_CMD_PROG_CSD : **stm32303c_eval_sd.h**
- SD_CMD_READ_MULT_BLOCK : **stm32303c_eval_sd.h**
- SD_CMD_READ_SINGLE_BLOCK : **stm32303c_eval_sd.h**
- SD_CMD_SD_ERASE_GRP_END : **stm32303c_eval_sd.h**
- SD_CMD_SD_ERASE_GRP_START : **stm32303c_eval_sd.h**
- SD_CMD_SEND_CID : **stm32303c_eval_sd.h**
- SD_CMD_SEND_CSD : **stm32303c_eval_sd.h**
- SD_CMD_SEND_OP_COND : **stm32303c_eval_sd.h**
- SD_CMD_SEND_STATUS : **stm32303c_eval_sd.h**
- SD_CMD_SEND_WRITE_PROT : **stm32303c_eval_sd.h**
- SD_CMD_SET_BLOCK_COUNT : **stm32303c_eval_sd.h**
- SD_CMD_SET_BLOCKLEN : **stm32303c_eval_sd.h**
- SD_CMD_SET_WRITE_PROT : **stm32303c_eval_sd.h**
- SD_CMD_STOP_TRANSMISSION : **stm32303c_eval_sd.h**
- SD_CMD_UNTAG_ERASE_GROUP : **stm32303c_eval_sd.h**

- SD_CMD_UNTAG_SECTOR : **stm32303c_eval_sd.h**
- SD_CMD_WRITE_MULT_BLOCK : **stm32303c_eval_sd.h**
- SD_CMD_WRITE_SINGLE_BLOCK : **stm32303c_eval_sd.h**
- SD_CS_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- SD_CS_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- SD_CS_GPIO_PORT : **stm32303c_eval.h**
- SD_CS_HIGH : **stm32303c_eval.h**
- SD_CS_LOW : **stm32303c_eval.h**
- SD_CS_PIN : **stm32303c_eval.h**
- SD_DETECT_EXTI_IRQn : **stm32303c_eval.h**
- SD_DETECT_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- SD_DETECT_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- SD_DETECT_GPIO_PORT : **stm32303c_eval.h**
- SD_DETECT_PIN : **stm32303c_eval.h**
- SD_DUMMY_BYTE : **stm32303c_eval_sd.c** , **stm32303c_eval.c**
- SD_NO_RESPONSE_EXPECTED : **stm32303c_eval.c** , **stm32303c_eval_sd.c**
- SD_NOT_PRESENT : **stm32303c_eval_sd.h**
- SD_PRESENT : **stm32303c_eval_sd.h**
- SD_START_DATA_MULTIPLE_BLOCK_READ : **stm32303c_eval_sd.h**
- SD_START_DATA_MULTIPLE_BLOCK_WRITE : **stm32303c_eval_sd.h**
- SD_START_DATA_SINGLE_BLOCK_READ : **stm32303c_eval_sd.h**
- SD_START_DATA_SINGLE_BLOCK_WRITE : **stm32303c_eval_sd.h**
- SD_STOP_DATA_MULTIPLE_BLOCK_WRITE : **stm32303c_eval_sd.h**
- SEL_JOY_EXTI_IRQn : **stm32303c_eval.h**
- SEL_JOY_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- SEL_JOY_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- SEL_JOY_GPIO_PORT : **stm32303c_eval.h**
- SEL_JOY_PIN : **stm32303c_eval.h**
- SET_INDEX : **stm32303c_eval.c**
- START_BYTE : **stm32303c_eval.c**

Generated on Wed May 31 2017 10:00:45 for STM32303C_EVAL BSP
User Manual by  1.7.6.1

STM32303C_EVAL BSP User Manual

- t -

- TSENSOR_I2C_ADDRESS : **stm32303c_eval_tsensor.h**
- TSENSOR_MAX_TRIALS : **stm32303c_eval_tsensor.h**

STM32303C_EVAL BSP User Manual

- u -

- UP_JOY_EXTI_IRQn : **stm32303c_eval.h**
- UP_JOY_GPIO_CLK_DISABLE : **stm32303c_eval.h**
- UP_JOY_GPIO_CLK_ENABLE : **stm32303c_eval.h**
- UP_JOY_GPIO_PORT : **stm32303c_eval.h**
- UP_JOY_PIN : **stm32303c_eval.h**

STM32303C_EVAL BSP User Manual

stm32303c_eval.c File Reference

This file provides a set of firmware functions to manage LEDs, push-button and COM port available on STM32303C-EVAL evaluation board (MB1019) RevC from STMicroelectronics. More...

```
#include "stm32303c_eval.h"
```

Go to the source code of this file.

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	**EEPROM_CMD_WREN** 0x06
#define	**EEPROM_CMD_WRDI** 0x04
#define	**EEPROM_CMD_RDSR** 0x05
#define	**EEPROM_CMD_WRSR** 0x01
#define	**EEPROM_CMD_WRITE** 0x02
#define	**EEPROM_CMD_READ** 0x03
#define	**EEPROM_WIP_FLAG** 0x01
#define	**__STM32303C_EVAL_BSP_VERSION_MAIN** (0x02) STM32303C EVAL BSP Driver version number V2.1.2.
#define	**__STM32303C_EVAL_BSP_VERSION_SUB1** (0x01)
#define	**__STM32303C_EVAL_BSP_VERSION_SUB2** (0x02)
#define	**__STM32303C_EVAL_BSP_VERSION_RC** (0x00)
#define	**__STM32303C_EVAL_BSP_VERSION**

Functions

static void	**I2Cx_Init** (void) Eval I2Cx Bus initialization.
static void	**I2Cx_WriteData** (uint16_t Addr, uint8_t Reg uint16_t RegSize, uint8_t Value) Write a value in a register of the device through BUS.
static HAL_StatusTypeDef	**I2Cx_WriteBuffer** (uint16_t Addr, uint8_t Re 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 uint16_t RegSize) Read a register of the device through BUS.
static HAL_StatusTypeDef	**I2Cx_ReadBuffer** (uint16_t Addr, uint8_t Re uint16_t RegSize, uint8_t *pBuffer, uint16_t Length) Reads multiple data on the BUS.
static HAL_StatusTypeDef	**I2Cx_IsDeviceReady** (uint16_t DevAddress uint32_t Trials) Checks if target device is ready for communication.
static void	**I2Cx_Error** (void) Eval I2Cx error treatment function.
static void	**I2Cx_MspInit** (I2C_HandleTypeDef *hi2c) Initializes I2C MSP.
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_ 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 peripherals used by the Audio Codec driver.
void	**AUDIO_IO_DeInit** (void) DeInitializes Audio low level.
void	**AUDIO_IO_Write** (uint16_t DevAddress, uint8_t Reg, uint8_t Value) Writes a single data on the Audio Codec.
uint8_t	**AUDIO_IO_Read** (uint16_t DevAddress, uint8_t Reg) Reads a single data from the Audio Codec.

void	**AUDIO_IO_Delay** (uint32_t Delay) Wait for loop in ms.
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	**EEPROM_SPI_IO_Init** (void) Initializes the EEPROM SPI and put it into StandBy State (Ready for data transfer).
void	**EEPROM_SPI_IO_WriteByte** (uint8_t Data) Write a byte on the EEPROM.
uint8_t	**EEPROM_SPI_IO_ReadByte** (void) Read a byte from the EEPROM.
HAL_StatusTypeDef	**EEPROM_SPI_IO_WriteData** (uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize)

	Write data to SPI EEPROM driver.
HAL_StatusTypeDef	**EEPROM_SPI_IO_ReadData** (uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Read data from SPI EEPROM driver.
HAL_StatusTypeDef	**EEPROM_SPI_IO_WaitEepromStandbySt** (void) Wait response from the SPI EEPROM.
void	**SD_IO_Init** (void) Initializes the SD Card and put it into StandE State (Ready for data transfer).
HAL_StatusTypeDef	**SD_IO_WriteCmd** (uint8_t Cmd, uint32_t A 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 STM32303C 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** Butto Returns the selected button state.
uint8_t	**BSP_JOY_Init** (**JOYMode_TypeDef** Joy_Mode) Configures all button of the joystick in GPIO or EXTI modes.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the current joystick status.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) Configures COM port.

Variables

GPIO_TypeDef *	**LED_PORT** [**LEDn**] LED variables.
const uint16_t	**LED_PIN** [**LEDn**]
GPIO_TypeDef *	**BUTTON_PORT** [**BUTTONn**] BUTTON variables.
const uint16_t	**BUTTON_PIN** [**BUTTONn**]
const uint16_t	**BUTTON_IRQn** [**BUTTONn**]
GPIO_TypeDef *	**JOY_PORT** [**JOYn**] JOYSTICK variables.
const uint16_t	**JOY_PIN** [**JOYn**]
const uint8_t	**JOY_IRQn** [**JOYn**]
USART_TypeDef *	**COM_USART** [**COMn**] = {**EVAL_COM1**} 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**}
const uint16_t	**COM_TX_AF** [**COMn**] = {**EVAL_COM1_TX_AF**}
const uint16_t	**COM_RX_AF** [**COMn**] = {**EVAL_COM1_RX_AF**}
uint32_t	**I2cxTimeout** = **EVAL_I2Cx_TIMEOUT_MAX** BUS variables.
I2C_HandleTypeDef	**heval_I2c**
uint32_t	**SpixTimeout** = **EVAL_SPIx_TIMEOUT_MAX**

static SPI_HandleTypeDef **heval_Spi**

Detailed Description

This file provides a set of firmware functions to manage LEDs, push-button and COM port available on STM32303C-EVAL evaluation board (MB1019) RevC from STMicroelectronics.

Author:

MCD Application Team

Attention:

© COPYRIGHT(c) 2016 STMicroelectronics

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

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

Definition in file **stm32303c_eval.c**.

STM32303C_EVAL BSP User Manual

stm32303c_eval.h File Reference

This file contains definitions for STM32303C_EVAL's Leds, push-buttons and COM port hardware resources. More...

```
#include "stm32f3xx_hal.h"
```

Go to the source code of this file.

Defines

#define **LEDn** 4

#define **LED1_PIN** GPIO_PIN_8

#define **LED1_GPIO_PORT** GPIOE

#define **LED1_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_CLK_E

#define **LED1_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_CLK_D

#define **LED2_PIN** GPIO_PIN_9

#define **LED2_GPIO_PORT** GPIOE

#define **LED2_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_CLK_E

#define **LED2_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_CLK_D

#define **LED3_PIN** GPIO_PIN_10

#define **LED3_GPIO_PORT** GPIOE

#define **LED3_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_CLK_E

#define **LED3_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_CLK_D

#define **LED4_PIN** GPIO_PIN_11

#define **LED4_GPIO_PORT** GPIOE

#define **LED4_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_CLK_E

#define **LED4_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_CLK_D

#define **LEDx_GPIO_CLK_ENABLE**(__LED__)

#define **LEDx_GPIO_CLK_DISABLE**(__LED__)

#define **JOYn** 5

#define **BUTTONn** 1 + **JOYn**

#define **KEY_BUTTON_PIN** GPIO_PIN_6

Key push-button.

#define **KEY_BUTTON_GPIO_PORT** GPIOE

#define **KEY_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE

#define **KEY_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_GPIO

#define **KEY_BUTTON_EXTI_IRQn** EXTI9_5_IRQn

#define **RIGHT_JOY_PIN** GPIO_PIN_6

Joystick Right push-button.

#define **RIGHT_JOY_GPIO_PORT** GPIOD

#define **RIGHT_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOD_

#define **RIGHT_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOD_

#define **RIGHT_JOY_EXTI_IRQn** EXTI9_5_IRQn

#define **LEFT_JOY_PIN** GPIO_PIN_5
Joystick Left push-button.

#define **LEFT_JOY_GPIO_PORT** GPIOB

#define **LEFT_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_C

#define **LEFT_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_C

#define **LEFT_JOY_EXTI_IRQn** EXTI9_5_IRQn

#define **UP_JOY_PIN** GPIO_PIN_7
Joystick Up push-button.

#define **UP_JOY_GPIO_PORT** GPIOE

#define **UP_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_CLK

#define **UP_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_CLK

#define **UP_JOY_EXTI_IRQn** EXTI9_5_IRQn

#define **DOWN_JOY_PIN** GPIO_PIN_5
Joystick Down push-button.

#define **DOWN_JOY_GPIO_PORT** GPIOD

#define **DOWN_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOD_

#define **DOWN_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOD_

#define **DOWN_JOY_EXTI_IRQn** EXTI9_5_IRQn

#define **SEL_JOY_PIN** GPIO_PIN_13
Joystick Sel push-button.

#define **SEL_JOY_GPIO_PORT** GPIOC

#define **SEL_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOC_CL

#define **SEL_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOC_CL

#define **SEL_JOY_EXTI_IRQn** EXTI15_10_IRQn

#define **BUTTONx_GPIO_CLK_ENABLE**(__BUTTON__)

#define **BUTTONx_GPIO_CLK_DISABLE**(__BUTTON__)

#define **JOYx_GPIO_CLK_ENABLE**(__JOY__)

#define **JOYx_GPIO_CLK_DISABLE**(__JOY__)

#define **COMn** 1

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

#define **EVAL_COM1_CLK_ENABLE**() __HAL_RCC_USART1_CLK_

```
#define EVAL_COM1_CLK_DISABLE()   __HAL_RCC_USART1_CLK
#define EVAL_COM1_TX_PIN   GPIO_PIN_4
#define EVAL_COM1_TX_GPIO_PORT   GPIOC
#define EVAL_COM1_TX_GPIO_CLK_ENABLE()   __HAL_RCC_GPI
#define EVAL_COM1_TX_GPIO_CLK_DISABLE()   __HAL_RCC_GP
#define EVAL_COM1_TX_AF   GPIO_AF7_USART1
#define EVAL_COM1_RX_PIN   GPIO_PIN_1
#define EVAL_COM1_RX_GPIO_PORT   GPIOE
#define EVAL_COM1_RX_GPIO_CLK_ENABLE()   __HAL_RCC_GPI
#define EVAL_COM1_RX_GPIO_CLK_DISABLE()   __HAL_RCC_GP
#define EVAL_COM1_RX_AF   GPIO_AF7_USART1
#define EVAL_COM1_IRQn   USART1_IRQn
#define COMx_CLK_ENABLE(__INDEX__)   do { if((__INDEX__) ==
EVAL_COM1_CLK_ENABLE();} while(0)
#define COMx_CLK_DISABLE(__INDEX__)   (((__INDEX__) == COM
EVAL_COM1_CLK_DISABLE() : 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 EVAL_I2Cx_SCL_PIN   GPIO_PIN_6
#define EVAL_I2Cx_SCL_GPIO_PORT   GPIOF
#define EVAL_I2Cx_SDA_PIN   GPIO_PIN_10
#define EVAL_I2Cx_SDA_GPIO_PORT   GPIOA
#define EVAL_I2Cx_SCL_SDA_AF   GPIO_AF4_I2C2
#define EVAL_I2Cx   I2C2
#define EVAL_I2Cx_CLK_ENABLE()   __HAL_RCC_I2C2_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_I2C2_FORCE_
#define EVAL_I2Cx_RELEASE_RESET()   __HAL_RCC_I2C2_RELE
```

#define **EVAL_I2Cx_EV_IRQn** I2C2_EV_IRQn

#define **EVAL_I2Cx_EV_IRQHandler** I2C2_EV_IRQHandler

#define **EVAL_I2Cx_ER_IRQn** I2C2_ER_IRQn

#define **EVAL_I2Cx_ER_IRQHandler** I2C2_ER_IRQHandler

#define **EVAL_I2Cx_TIMEOUT_MAX** 1000

#define **EVAL_SPIx** SPI2
Definition for SPI Interface pins (SPI2 used)

#define **EVAL_SPIx_CLK_ENABLE**() __HAL_RCC_SPI2_CLK_ENA

#define **EVAL_SPIx_CLK_DISABLE**() __HAL_RCC_SPI2_CLK_DIS

#define **EVAL_SPIx_FORCE_RESET**() __HAL_RCC_SPI2_FORCE_

#define **EVAL_SPIx_RELEASE_RESET**() __HAL_RCC_SPI2_RELE

#define **EVAL_SPIx_SCK_AF** GPIO_AF5_SPI2

#define **EVAL_SPIx_SCK_GPIO_PORT** GPIOF

#define **EVAL_SPIx_SCK_PIN** GPIO_PIN_9

#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_AF** GPIO_AF5_SPI2

#define **EVAL_SPIx_MISO_MOSI_GPIO_PORT** GPIOB

#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_14

#define **EVAL_SPIx_MOSI_PIN** GPIO_PIN_15

#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_PO** GPIO_PIN_SET)

#define **LCD_NCS_PIN** GPIO_PIN_0 /* PE.00 */
LCD Control Interface pins.

#define **LCD_NCS_GPIO_PORT** GPIOE

#define **LCD_NCS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_CL

#define **LCD_NCS_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_CL

#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_15 /* PE.15 */ SD card Control pins.
#define	**SD_CS_GPIO_PORT** GPIOE
#define	**SD_CS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_CLK_
#define	**SD_CS_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_CLK_
#define	**SD_DETECT_PIN** GPIO_PIN_6 /* PC.06 */ SD Detect Interface pins.
#define	**SD_DETECT_GPIO_PORT** GPIOC
#define	**SD_DETECT_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOC_
#define	**SD_DETECT_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOC_
#define	**SD_DETECT_EXTI_IRQn** EXTI9_5_IRQn
#define	**EEPROM_CS_LOW**() HAL_GPIO_WritePin(**EEPROM_CS_G** **EEPROM_CS_PIN**, GPIO_PIN_RESET)
#define	**EEPROM_CS_HIGH**() HAL_GPIO_WritePin(**EEPROM_CS_C** **EEPROM_CS_PIN**, GPIO_PIN_SET)
#define	**EEPROM_CS_PIN** GPIO_PIN_7 EEPROM Control Interface pins.
#define	**EEPROM_CS_GPIO_PORT** GPIOD
#define	**EEPROM_CS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOD_
#define	**EEPROM_CS_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOD

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_KEY = 0, **BUTTON_SEL** = 1, **BUTTON_LEFT** = 2, **BUTTON_RIGHT** = 3,
	BUTTON_DOWN = 4, **BUTTON_UP** = 5
	}
	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 } COM Types Definition. More...

Functions

uint32_t	**BSP_GetVersion** (void) This method returns the STM32303C EVAL BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LED GPIO.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** Button_Mode) Configures push button GPIO and EXTI Line.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
uint8_t	**BSP_JOY_Init** (**JOYMode_TypeDef** Joy_Mode) Configures all button of the joystick in GPIO or EXTI modes.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the current joystick status.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) Configures COM port.

Detailed Description

This file contains definitions for STM32303C_EVAL's Leds, push-buttons and COM port hardware resources.

Author:

MCD Application Team

Attention:

© COPYRIGHT(c) 2016 STMicroelectronics

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

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

Definition in file **stm32303c_eval.h**.

STM32303C_EVAL BSP User Manual

stm32303c_eval_audio.c File Reference

This file provides the Audio driver for the STM32303C_EVAL evaluation board(MB1019). More...

```
#include "stm32303c_eval_audio.h"
```

Go to the source code of this file.

Functions

static void	**I2Sx_MspInit** (void) AUDIO OUT I2S MSP Init.
static **AUDIO_StatusTypeDef**	**I2Sx_Init** (uint32_t AudioFreq) Initializes the Audio Codec audio interfa
uint8_t	**BSP_AUDIO_OUT_Init** (uint16_t Outpu uint8_t Volume, uint32_t AudioFreq) Configure the audio peripherals.
uint8_t	**BSP_AUDIO_OUT_Play** (uint16_t *pBu uint32_t Size) Starts playing audio stream from a data a determined size.
uint8_t	**BSP_AUDIO_OUT_ChangeBuffer** (uint *pData, uint16_t Size) Sends n-Bytes on the I2S interface.
uint8_t	**BSP_AUDIO_OUT_Pause** (void) This function Pauses the audio file strea
uint8_t	**BSP_AUDIO_OUT_Resume** (void) This function Resumes the audio file stre
uint8_t	**BSP_AUDIO_OUT_Stop** (uint32_t Opti Stops audio playing and Power down the Codec.
uint8_t	**BSP_AUDIO_OUT_SetVolume** (uint8_t Controls the current audio volume level.
uint8_t	**BSP_AUDIO_OUT_SetMute** (uint32_t C Enables or disables the MUTE mode by
uint8_t	**BSP_AUDIO_OUT_SetOutputMode** (u Output) Switch dynamically (while audio file is pl output target (speaker or headphone).
uint8_t	**BSP_AUDIO_OUT_SetFrequency** (uint AudioFreq)

	Update the audio frequency.
void	**HAL_I2S_TxCpltCallback** (I2S_Handle *hi2s) Tx Transfer completed callbacks.
void	**HAL_I2S_TxHalfCpltCallback** (I2S_HandleTypeDef *hi2s) Tx Transfer Half completed callbacks.
void	**HAL_I2S_ErrorCallback** (I2S_HandleTy *hi2s) I2S error callbacks.
__weak void	**BSP_AUDIO_OUT_TransferComplete_** (void) Manages the DMA full Transfer complete
__weak void	**BSP_AUDIO_OUT_HalfTransfer_CallB** (void) Manages the DMA Half Transfer comple
__weak void	**BSP_AUDIO_OUT_Error_CallBack** (vo Audio OUT Error callback function.

Variables

static AUDIO_DrvTypeDef *	**pAudioDrv** = NULL
I2S_HandleTypeDef	**hAudioOutI2s**

Detailed Description

This file provides the Audio driver for the STM32303C_EVAL evaluation board(MB1019).

Author:

MCD Application Team

Attention:

© COPYRIGHT(c) 2016 STMicroelectronics

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

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

Definition in file **stm32303c_eval_audio.c**.

STM32303C_EVAL BSP User Manual

stm32303c_eval_audio.h File Reference

This file contains all the functions prototypes for the **stm32303c_eval_audio.c** driver. More...

```
#include "../Components/cs42l52/cs42l52.h" #include
"stm32303c_eval.h"
```

Go to the source code of this file.

Defines

#define	**AUDIO_I2C_ADDRESS** 0x94
#define	**I2Sx** SPI3
#define	**I2Sx_CLK_ENABLE**() __HAL_RCC_SPI3_CLK_ENABLE()
#define	**I2Sx_CLK_DISABLE**() __HAL_RCC_SPI3_CLK_DISABLE()
#define	**I2Sx_FORCE_RESET**() __HAL_RCC_SPI3_FORCE_RESET
#define	**I2Sx_RELEASE_RESET**() __HAL_RCC_SPI3_RELEASE_R
#define	**I2Sx_WS_PIN** GPIO_PIN_4
#define	**I2Sx_MCK_PIN** GPIO_PIN_9
#define	**I2Sx_SCK_PIN** GPIO_PIN_10
#define	**I2Sx_DIN_PIN** GPIO_PIN_12
#define	**I2Sx_WS_GPIO_PORT** GPIOA
#define	**I2Sx_MCK_GPIO_PORT** GPIOA
#define	**I2Sx_SCK_DIN_GPIO_PORT** GPIOC
#define	**I2Sx_MCK_WS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIO
#define	**I2Sx_MCK_WS_GPIO_CLK_DISABLE**() __HAL_RCC_GPIC
#define	**I2Sx_WS_AF** GPIO_AF6_SPI3
#define	**I2Sx_MCK_AF** GPIO_AF5_SPI3
#define	**I2Sx_SCK_DIN_GPIO_CLK_ENABLE**() __HAL_RCC_GPIO
#define	**I2Sx_SCK_DIN_GPIO_CLK_DISABLE**() __HAL_RCC_GPIC
#define	**I2Sx_SCK_DIN_AF** GPIO_AF6_SPI3
#define	**I2Sx_DMAx_CLK_ENABLE**() __HAL_RCC_DMA2_CLK_EN
#define	**I2Sx_DMAx_CLK_DISABLE**() __HAL_RCC_DMA2_CLK_DI
#define	**I2Sx_DMAx_CHANNEL** DMA2_Channel2
#define	**I2Sx_DMAx_IRQ** DMA2_Channel2_IRQn
#define	**I2Sx_DMAx_PERIPH_DATA_SIZE** DMA_PDATAALIGN_HAL
#define	**I2Sx_DMAx_MEM_DATA_SIZE** DMA_MDATAALIGN_HALFV
#define	**DMA_MAX_SZE** 0xFFFF
#define	**AUDIO_OUT_IRQ_PREPRIO** ((uint32_t)0x0E) /* Select the p level(0 is the highest) */
#define	**AUDIO_OUT_IRQ_SUBPRIO** 0 /* Select the sub-priority leve
#define	**AUDIODATA_SIZE** 2 /* 16-bits audio data size */

```
#define  DMA_MAX(_X_)  (((_X_) <= DMA_MAX_SZE)? (_X_):DMA_M
```

Enumerations

enum	**AUDIO_StatusTypeDef** { **AUDIO_OK** = 0x00, **AUDIO_ERROR** = 0x01, **AUDIO_TIMEOUT** = 0x02 }

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.
uint8_t	**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_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)
Audio OUT Error callback function.

Detailed Description

This file contains all the functions prototypes for the **stm32303c_eval_audio.c** driver.

Author:

MCD Application Team

Attention:

© COPYRIGHT(c) 2016 STMicroelectronics

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

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

Definition in file **stm32303c_eval_audio.h**.

STM32303C_EVAL BSP User Manual

stm32303c_eval_eeprom.c File Reference

This file provides a set of functions needed to manage a M24LR64 or M24M01 I2C EEPROM memory, or a M95M01 SPI EEPROM memory. More...

```
#include "stm32303c_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.
static uint32_t	**EEPROM_SPI_Init** (void) Initializes peripherals used by the SPI EEPROM driver.
static uint32_t	**EEPROM_SPI_ReadBuffer** (uint8_t *pBuffer, uint16_t ReadAddr, uint32_t *NumByteToRead) Reads a block of data from the SPI EEPROM.
static uint32_t	**EEPROM_SPI_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_SPI_WaitEepromStandbyState** (void) Wait for EEPROM SPI 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**
__IO uint8_t	**EEPROMDataWrite**
static **EEPROM_DrvTypeDef** *	**EEPROM_SelectedDevice** = 0
EEPROM_DrvTypeDef	**EEPROM_I2C_Drv**
EEPROM_DrvTypeDef	**EEPROM_SPI_Drv**

Detailed Description

This file provides a set of functions needed to manage a M24LR64 or M24M01 I2C EEPROM memory, or a M95M01 SPI EEPROM memory.

Author:

MCD Application Team

==========

Notes:

- This driver is intended for STM32F30x families devices only.
- The I2C EEPROM memory (M24LR64) is available on separate daughter board ANT7-M24LR-A, which is provided with the STM32F303C EVAL board. To use this driver with M24LR64, you have to connect the ANT7-M24LR-A to CN1 connector of STM32F303C EVAL board. Jumper JP5 and JP6 needs to be set in I2C2 position. Jumper JP8 (E2P WC) needs to be set.

- The I2C EEPROM memory (M24M01) is mounted on the STM32F303C EVAL board. Jumper JP5 and JP6 needs to be set in I2C2_F position. Jumper JP8 (E2P WC) needs to be set.

- The SPI EEPROM memory (M95M01) is available directly on STM32F303C EVAL board.

==========

It implements a high level communication layer for read and write from/to this memory. The needed STM32F30x hardware resources (I2C, SPI and GPIO) are defined in **stm32303c_eval.h** file, and the initialization is performed depending of EEPROMs in **EEPROM_I2C_IO_Init()** or **EEPROM_SPI_IO_Init()** functions declared in **stm32303c_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()** or **EEPROM_SPI_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 M24M01 EEPROM | +----------+----------+----------+ | STM32F30x I2C Pins | EEPROM | Pin | +---------- ----------+----------+----------+ | . | NC | 1 | | . | E1(VDD) | 2 (3.3V) | | . | E2(GND) | 3 (0V) | | . | VSS | 4 (0V) | | EEPROM_I2C_SDA_PIN/ SDA | SDA | 5 | | EEPROM_I2C_SCL_PIN/ SCL | SCL | 6 | | . | /WC(VSS)| 7 (0V) | | . | VDD | 8 (3.3V) | +----------+----------+---------- ----------+ +----------+ | Pin assignment for M24LR64 EEPROM | +---------- ---+----------+----------+ | STM32F30x I2C Pins | EEPROM | Pin | +----------+----------+----------+ | . | E0(GND) | 1 (0V) | | . | AC0 | 2 | | . | AC1 | 3 | | . | VSS | 4 (0V) | | EEPROM_I2C_SDA_PIN/ SDA | SDA | 5 | | EEPROM_I2C_SCL_PIN/ SCL | SCL | 6 | | . | E1(GND) | 7 (0V) | | . | VDD | 8 (3.3V) | +----------+----------+---------- ----------+

+----------+ | Pin assignment for M95M01 EEPROM | +----------+----------+---------- ----------+ | STM32F30x SPI Pins | EEPROM | Pin | +---------- ----------+----------+----------+ | EEPROM_CS_PIN | ChipSelect| 1 (/S) | | EEPROM_SPI_MISO_PIN/ MISO | DataOut | 2 (Q) | | EEPROM_SPI_WP_PIN | WR Protect| 3 (/W) | | . | GND | 4 (0 V) | | EEPROM_SPI_MOSI_PIN/ MOSI | DataIn | 5 (D) | | EEPROM_SPI_SCK_PIN/ SCLK | Clock | 6 (C) | | . | Hold | 7 (/HOLD)| | . | VCC | 8 (3.3 V)| +----------+----------+----------+

Attention:

© COPYRIGHT(c) 2016 STMicroelectronics

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

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

Definition in file **stm32303c_eval_eeprom.c**.

STM32303C_EVAL BSP User Manual

stm32303c_eval_eeprom.h File Reference

This file contains all the functions prototypes for the **stm32303c_eval_eeprom.c** firmware driver. More...

```
#include "stm32303c_eval.h"
```

Go to the source code of this file.

Data Structures

struct	**EEPROM_DrvTypeDef**

Defines

#define	**EEPROM_ADDRESS_M24M01** 0xA4 /* EEPROM M24M01-HR used */
#define	**EEPROM_ADDRESS_M24LR64_A01** 0xA0 /* RF EEPROM ANT7-M24LR-A01 used */
#define	**EEPROM_ADDRESS_M24LR64_A02** 0xA6 /* RF EEPROM ANT7-M24LR-A02 used */
#define	**EEPROM_PAGESIZE_M24M01** 28 /* EEPROM M24M01-HR used */
#define	**EEPROM_PAGESIZE_M24LR64** 4 /* RF EEPROM ANT7-M24LR-A used */
#define	**EEPROM_PAGESIZE_M95M01** 256 /* EEPROM M95M01 used */
#define	**EEPROM_OK** 0
#define	**EEPROM_FAIL** 1
#define	**EEPROM_TIMEOUT** 2
#define	**BSP_EEPROM_M24LR64** 1 /* RF I2C EEPROM M24LR64 */
#define	**BSP_EEPROM_M24M01** 2 /* I2C EEPROM M24M01 */
#define	**BSP_EEPROM_M95M01** 3 /* SPI EEPROM M95M01 */
#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.

void	**EEPROM_SPI_IO_Init** (void) Initializes the EEPROM SPI and put it into StandBy State (Ready for data transfer).
void	**EEPROM_SPI_IO_WriteByte** (uint8_t Data) Write a byte on the EEPROM.
uint8_t	**EEPROM_SPI_IO_ReadByte** (void) Read a byte from the EEPROM.
HAL_StatusTypeDef	**EEPROM_SPI_IO_WriteData** (uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Write data to SPI EEPROM driver.
HAL_StatusTypeDef	**EEPROM_SPI_IO_ReadData** (uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Read data from SPI EEPROM driver.
HAL_StatusTypeDef	**EEPROM_SPI_IO_WaitEepromStandbyState** (void) Wait response from the SPI EEPROM.
void	**EEPROM_SPI_IO_WriteDummy** (void)

Detailed Description

This file contains all the functions prototypes for the **stm32303c_eval_eeprom.c** firmware driver.

Author:

MCD Application Team

Attention:

© COPYRIGHT(c) 2016 STMicroelectronics

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

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

Definition in file **stm32303c_eval_eeprom.h**.

STM32303C_EVAL BSP User Manual

stm32303c_eval_lcd.c File Reference

This file includes the driver for Liquid Crystal Display modules mounted on STM32303C-EVAL evaluation board. More...

```
#include "stm32303c_eval_lcd.h" #include
"../../../Utilities/Fonts/fonts.h"
#include "../../../Utilities/Fonts/font24.c"
#include "../../../Utilities/Fonts/font20.c"
#include "../../../Utilities/Fonts/font16.c"
#include "../../../Utilities/Fonts/font12.c"
#include "../../../Utilities/Fonts/font8.c"
```

Go to the source code of this file.

Defines

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

Functions

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

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

	Draws a bitmap picture loaded in the internal Flash (32 bpp).
void	**BSP_LCD_FillRect** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Draws a full rectangle.
void	**BSP_LCD_FillCircle** (uint16_t Xpos, uint16_t Ypos, uint16_t Radius) Draws a full circle.
void	**BSP_LCD_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 modules mounted on STM32303C-EVAL evaluation board.

Author:

MCD Application Team

Attention:

© COPYRIGHT(c) 2016 STMicroelectronics

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

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

Definition in file **stm32303c_eval_lcd.c**.

STM32303C_EVAL BSP User Manual

Data Structures | Defines | Typedefs | Enumerations | Functions

stm32303c_eval_lcd.h File Reference

This file contains all the functions prototypes for the **stm32303c_eval_lcd.c** driver. More...

```
#include "stm32303c_eval.h" #include
"../Components/hx8347d/hx8347d.h"
#include "../Components/spfd5408/spfd5408.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** pPoints, uint16_t PointCount) Draws an poly-line (between many points).
void	**BSP_LCD_DrawEllipse** (int Xpos, int Ypos, int XRadius, int YRadius) Draws an ellipse on LCD.
void	**BSP_LCD_DrawBitmap** (uint16_t Xpos, uint16_t Ypos, uint8_t *pBmp) Draws a bitmap picture loaded in the internal Flash (32 bpp).
void	**BSP_LCD_FillRect** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Draws a full rectangle.
void	**BSP_LCD_FillCircle** (uint16_t Xpos, uint16_t Ypos, uint16_t Radius) Draws a full circle.
void	**BSP_LCD_FillEllipse** (int Xpos, int Ypos, int XRadius, int YRadius) Draws a full ellipse.

void	**BSP_LCD_DisplayOff** (void) Disables the display.
void	**BSP_LCD_DisplayOn** (void) Enables the display.

Detailed Description

This file contains all the functions prototypes for the **stm32303c_eval_lcd.c** driver.

Author:

MCD Application Team

Attention:

© COPYRIGHT(c) 2016 STMicroelectronics

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

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

Definition in file **stm32303c_eval_lcd.h**.

STM32303C_EVAL BSP User Manual

stm32303c_eval_sd.c File Reference

This file provides a set of functions needed to manage the SPI SD Card memory mounted on STM32303C-EVAL board. It implements a high level communication layer for read and write from/to this memory. The needed STM32F30x hardware resources (SPI and GPIO) are defined in **stm32303c_eval.h** file, and the initialization is performed in SD_LowLevel_Init() function declared in **stm32303c_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and SD_LowLevel_Init() function. More...

```
#include "stm32303c_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 STM32303C-EVAL board. It implements a high level communication layer for read and write from/to this memory. The needed STM32F30x hardware resources (SPI and GPIO) are defined in **stm32303c_eval.h** file, and the initialization is performed in SD_LowLevel_Init() function declared in **stm32303c_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and SD_LowLevel_Init() function.

Author:

MCD Application Team +----------+ | Pin assignment | +----------+ | STM32F30x SPI Pins | SD | Pin | +----------+ | SD_SPI_CS_PIN | ChipSelect | 1 | | SD_SPI_MOSI_PIN / MOSI | DataIn | 2 | | | GND | 3 (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:

© COPYRIGHT(c) 2016 STMicroelectronics

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

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

Definition in file **stm32303c_eval_sd.c**.

STM32303C_EVAL BSP User Manual

Data Structures | Defines | Enumerations | Functions

stm32303c_eval_sd.h File Reference

This file contains the common defines and functions prototypes for the **stm32303c_eval_sd.c** driver. More...

```
#include "stm32303c_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_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 **stm32303c_eval_sd.c** driver.

Author:

MCD Application Team

Attention:

© COPYRIGHT(c) 2016 STMicroelectronics

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

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

Definition in file **stm32303c_eval_sd.h**.

STM32303C_EVAL BSP User Manual

stm32303c_eval_tsensor.c File Reference

This file provides a set of functions needed to manage the I2C TS751 temperature sensor mounted on STM32303C-EVAL board . It implements a high level communication layer for read and write from/to this sensor. The needed STM323F30x hardware resources (I2C and GPIO) are defined in **stm32303c_eval.h** file, and the initialization is performed in **TSENSOR_IO_Init()** function declared in **stm32303c_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **TSENSOR_IO_Init()** function. More...

```
#include "stm32303c_eval_tsensor.h"
```

Go to the source code of this file.

Functions

uint32_t	**BSP_TSENSOR_Init** (void) Initializes peripherals used by the I2C Temperature Sensor driver.
uint8_t	**BSP_TSENSOR_ReadStatus** (void) Returns the Temperature Sensor status.
uint16_t	**BSP_TSENSOR_ReadTemp** (void) Read Temperature register of TS751.

Variables

static TSENSOR_DrvTypeDef * **tsensor_drv**

Detailed Description

This file provides a set of functions needed to manage the I2C TS751 temperature sensor mounted on STM32303C-EVAL board . It implements a high level communication layer for read and write from/to this sensor. The needed STM323F30x hardware resources (I2C and GPIO) are defined in **stm32303c_eval.h** file, and the initialization is performed in **TSENSOR_IO_Init()** function declared in **stm32303c_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **TSENSOR_IO_Init()** function.

Author:

MCD Application Team +---------- ----------+ | Pin assignment | +---------- ----------+----------+ | STM32F30x I2C Pins | STTS751 | Pin | +---------- ----------+----------+ | . | Addr/Therm | 1 | | . | GND | 2 (0V) | | . | VDD | 3 (3.3V)| | TSENSOR_I2C_SCL_PIN/ SCL | SCL | 4 | | TSENSOR_I2C_SMBUSALERT_PIN/ SMBUS ALERT| SMBUS ALERT| 5 | | TSENSOR_I2C_SDA_PIN/ SDA | SDA | 6 | +---------- ----------+----------+

Attention:

© COPYRIGHT(c) 2016 STMicroelectronics

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

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

Definition in file **stm32303c_eval_tsensor.c**.

STM32303C_EVAL BSP User Manual

stm32303c_eval_tsensor.h File Reference

This file contains all the functions prototypes for the **stm32303c_eval_tsensor.c** firmware driver. More...

```
#include "stm32303c_eval.h" #include
"../Components/stts751/stts751.h"
```

Go to the source code of this file.

Defines

#define **TSENSOR_I2C_ADDRESS** 0x90
#define **TSENSOR_MAX_TRIALS** 50

Enumerations

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

Functions

uint32_t	**BSP_TSENSOR_Init** (void) Initializes peripherals used by the I2C Temperature Sensor driver.
uint8_t	**BSP_TSENSOR_ReadStatus** (void) Returns the Temperature Sensor status.
uint16_t	**BSP_TSENSOR_ReadTemp** (void) Read Temperature register of TS751.

Detailed Description

This file contains all the functions prototypes for the **stm32303c_eval_tsensor.c** firmware driver.

Author:
MCD Application Team

Attention:

© COPYRIGHT(c) 2016 STMicroelectronics

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

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

Definition in file **stm32303c_eval_tsensor.h**.

STM32303C_EVAL BSP User Manual

Modules

Here is a list of all modules:

- BSP
  - STM32303C-EVAL
    - STM32303C-EVAL Common
      - Bus Operation functions
      - Link Operation functions
      - Private Constants
      - Private Variables
      - Exported Types
      - Exported Constants
        - STM32303C-EVAL LED
        - STM32303C-EVAL BUTTON
        - STM32303C-EVAL COM
        - STM32303C-EVAL COMPONENT
      - Exported Functions
    - STM32303C_EVAL AUDIO
      - Private Types
      - Private Constants
      - Private Macros
      - Private Variables
      - Private Functions
      - Exported Types
      - Exported Constants
      - Exported Variables
      - Exported Macros
      - Exported Functions

- STM32303C-EVAL EEPROM
  - Private Types
  - Private Variables
  - Exported Types
  - Exported Constants
  - Exported Functions
  - STM32303C_EVAL_EEPROM_Private_Functions
- STM32303C-EVAL LCD
  - Private Defines
  - Private Macros
  - Private Variables
  - Private Functions
  - Exported Types
  - Exported Constants
  - Exported Functions
- STM32303C-EVAL SD
  - Types Definitions
  - Private Constants
  - Private Variables
  - Exported Types
  - Exported Constants
  - Exported Macro
  - Exported Functions
  - Private Functions
- STM32303C-EVAL TSENSOR
  - Private Variables
  - Exported Types
  - Exported Constants
  - Exported Functions
  - Private Functions

STM32303C_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

STM32303C_EVAL BSP User Manual

File List

Here is a list of all files with brief descriptions:

stm32303c_eval.c [code]	This file provides a set of firmware functions to manage LEDs, push-button and COM port available on STM32303C-EVAL evaluation board (MB1019) RevC from STMicroelectronics
stm32303c_eval.h [code]	This file contains definitions for STM32303C_EVAL's Leds, push-buttons and COM port hardware resources
stm32303c_eval_audio.c [code]	This file provides the Audio driver for the STM32303C_EVAL evaluation board(MB1019)
stm32303c_eval_audio.h [code]	This file contains all the functions prototypes for the **stm32303c_eval_audio.c** driver
stm32303c_eval_eeprom.c [code]	This file provides a set of functions needed to manage a M24LR64 or M24M01 I2C EEPROM memory, or a

	M95M01 SPI EEPROM memory
stm32303c_eval_eeprom.h [code]	This file contains all the functions prototypes for the **stm32303c_eval_eeprom.c** firmware driver
stm32303c_eval_lcd.c [code]	This file includes the driver for Liquid Crystal Display modules mounted on STM32303C-EVAL evaluation board
stm32303c_eval_lcd.h [code]	This file contains all the functions prototypes for the **stm32303c_eval_lcd.c** driver
stm32303c_eval_sd.c [code]	This file provides a set of functions needed to manage the SPI SD Card memory mounted on STM32303C-EVAL board. It implements a high level communication layer for read and write from/to this memory. The needed STM32F30x hardware resources (SPI and GPIO) are defined in **stm32303c_eval.h** file, and the initialization is performed in SD_LowLevel_Init() function declared in **stm32303c_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and SD_LowLevel_Init() function

stm32303c_eval_sd.h [code]	This file contains the common defines and functions prototypes for the **stm32303c_eval_sd.c** driver
stm32303c_eval_tsensor.c [code]	This file provides a set of functions needed to manage the I2C TS751 temperature sensor mounted on STM32303C-EVAL board . It implements a high level communication layer for read and write from/to this sensor. The needed STM323F30x hardware resources (I2C and GPIO) are defined in **stm32303c_eval.h** file, and the initialization is performed in **TSENSOR_IO_Init()** function declared in **stm32303c_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **TSENSOR_IO_Init()** function
stm32303c_eval_tsensor.h [code]	This file contains all the functions prototypes for the **stm32303c_eval_tsensor.c** firmware driver

STM32303C_EVAL BSP User Manual

Directories

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

- Firmware
  - Drivers
    - BSP
      - STM32303C_EVAL

STM32303C_EVAL BSP User Manual

STM32303C_EVAL AUDIO

This file includes the low layer audio driver available on STM32303C_EVAL evaluation board(MB1019). More...

Modules

- Private Types
- Private Constants
- Private Macros
- Private Variables
- Private Functions
- Exported Types
- Exported Constants
- Exported Variables
- Exported Macros
- Exported Functions

Detailed Description

This file includes the low layer audio driver available on STM32303C_EVAL evaluation board(MB1019).

STM32303C_EVAL BSP User Manual

Data Structure Index

E | L | P | S

E

EEPROM_DrvTypeDef

L

LCD_DrawPropTypeDef

P

Point

S

SD_Card

E | L | P | S

STM32303C_EVAL BSP User Manual

Exported Types

STM32303C-EVAL EEPROM

Data Structures

struct	**EEPROM_DrvTypeDef**

STM32303C_EVAL BSP User Manual

stm32303c_eval_eeprom.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32303c_eval_eeprom.h
00004   * @author  MCD Application Team
00005   * @brief   This file contains all the functions prototypes for
00006   *          the stm32303c_eval_eeprom.c firmware driver.
00007   ******************************************************************************
00008   * @attention
00009   *
00010   * <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>
00011   *
00012   * Redistribution and use in source and binary forms, with or without modification,
00013   * are permitted provided that the following conditions are met:
00014   *   1. Redistributions of source code must retain the above copyright notice,
```

```
00015   *      this list of conditions and the fol
lowing disclaimer.
00016   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00017   *      this list of conditions and the fol
lowing disclaimer in the documentation
00018   *      and/or other materials provided wit
h the distribution.
00019   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00020   *      may be used to endorse or promote p
roducts derived from this software
00021   *      without specific prior written perm
ission.
00022   *
00023   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00024   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00025   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00026   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00027   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00028   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
00029   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00030   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00031   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00032   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00033   *
00034   ****************************************
```

```
**********************************
00035   */
00036
00037 /* Define to prevent recursive inclusion ---
-------------------------------------*/
00038 #ifndef __STM32303C_EVAL_EEPROM_H
00039 #define __STM32303C_EVAL_EEPROM_H
00040
00041 #ifdef __cplusplus
00042  extern "C" {
00043 #endif
00044
00045 /* Includes ------------------------------
-------------------------------------*/
00046 #include "stm32303c_eval.h"
00047
00048 /** @addtogroup BSP
00049   * @{
00050   */
00051
00052 /** @addtogroup STM32303C_EVAL
00053   * @{
00054   */
00055
00056 /** @defgroup STM32303C_EVAL_EEPROM STM32303
C-EVAL EEPROM
00057   * @{
00058   */
00059
00060 /** @defgroup STM32303C_EVAL_EEPROM_Private_
Variables Private Variables
00061   * @{
00062   */
00063 /**
00064   * @}
00065   */
00066
```

```
00067 /** @defgroup STM32303C_EVAL_EEPROM_Exported
_Types Exported Types
00068   * @{
00069   */
00070 typedef struct
00071 {
00072   uint32_t  (*Init)(void);
00073   uint32_t  (*ReadBuffer)(uint8_t* , uint16_
t , uint32_t* );
00074   uint32_t  (*WritePage)(uint8_t* , uint16_t
 , uint32_t* );
00075 }EEPROM_DrvTypeDef;
00076 /**
00077   * @}
00078   */
00079
00080 /** @defgroup STM32303C_EVAL_EEPROM_Exported
_Constants Exported Constants
00081   * @{
00082   */
00083 /* EEPROMs hardware address and page size */

00084 #define EEPROM_ADDRESS_M24M01           0xA4
    /* EEPROM M24M01-HR used */
00085 #define EEPROM_ADDRESS_M24LR64_A01      0xA0
    /* RF EEPROM ANT7-M24LR-A01 used */
00086 #define EEPROM_ADDRESS_M24LR64_A02      0xA6
    /* RF EEPROM ANT7-M24LR-A02 used */
00087
00088 #define EEPROM_PAGESIZE_M24M01          28
    /* EEPROM M24M01-HR used */
00089 #define EEPROM_PAGESIZE_M24LR64         4
    /* RF EEPROM ANT7-M24LR-A used */
00090 #define EEPROM_PAGESIZE_M95M01          256
    /* EEPROM M95M01 used */
00091
00092 /* EEPROM BSP return values */
```

```
00093 #define EEPROM_OK                       0
00094 #define EEPROM_FAIL                     1
00095 #define EEPROM_TIMEOUT                  2
00096
00097 /* EEPROM BSP devices definition list suppor
ted */
00098 #define BSP_EEPROM_M24LR64              1
    /* RF I2C EEPROM M24LR64 */
00099 #define BSP_EEPROM_M24M01               2
    /* I2C EEPROM M24M01 */
00100 #define BSP_EEPROM_M95M01               3
    /* SPI EEPROM M95M01 */
00101
00102 /* Maximum number of trials for EEPROM_I2C_W
aitEepromStandbyState() function */
00103 #define EEPROM_MAX_TRIALS               300
00104 /**
00105   * @}
00106   */
00107
00108 /** @defgroup STM32303C_EVAL_EEPROM_Exported
_Functions Exported Functions
00109   * @{
00110   */
00111 uint32_t  BSP_EEPROM_Init(void);
00112 void      BSP_EEPROM_SelectDevice(uint8_t De
viceID);
00113 uint32_t  BSP_EEPROM_ReadBuffer(uint8_t* pBu
ffer, uint16_t ReadAddr, uint32_t* NumByteToRead);
00114 uint32_t  BSP_EEPROM_WriteBuffer(uint8_t* pB
uffer, uint16_t WriteAddr, uint32_t NumByteToWrite
);
00115
00116 /* USER Callbacks: This function is declared
 as __weak in EEPROM driver and
00117    should be implemented into user applicati
on.
```

```
00118    BSP_EEPROM_TIMEOUT_UserCallback() functio
n is called whenever a timeout condition
00119    occure during communication (waiting on a
n event that doesn't occur, bus
00120    errors, busy devices ...). */
00121 void BSP_EEPROM_TIMEOUT_UserCallback(void);
00122
00123
00124 /* Link functions for I2C EEPROM peripheral
*/
00125 void                    EEPROM_I2C_IO_Init(v
oid);
00126 HAL_StatusTypeDef       EEPROM_I2C_IO_WriteD
ata(uint16_t DevAddress, uint16_t MemAddress, uint
8_t* pBuffer, uint32_t BufferSize);
00127 HAL_StatusTypeDef       EEPROM_I2C_IO_ReadDa
ta(uint16_t DevAddress, uint16_t MemAddress, uint8
_t* pBuffer, uint32_t BufferSize);
00128 HAL_StatusTypeDef       EEPROM_I2C_IO_IsDevi
ceReady(uint16_t DevAddress, uint32_t Trials);
00129
00130 /* Link functions for EEPROM peripheral over
 SPI */
00131 void                    EEPROM_SPI_IO_Init(v
oid);
00132 void                    EEPROM_SPI_IO_WriteB
yte(uint8_t Data);
00133 uint8_t                 EEPROM_SPI_IO_ReadBy
te(void);
00134 HAL_StatusTypeDef       EEPROM_SPI_IO_WriteD
ata(uint16_t MemAddress, uint8_t* pBuffer, uint32_
t BufferSize);
00135 HAL_StatusTypeDef       EEPROM_SPI_IO_ReadDa
ta(uint16_t MemAddress, uint8_t* pBuffer, uint32_t
 BufferSize);
00136 HAL_StatusTypeDef       EEPROM_SPI_IO_WaitEe
promStandbyState(void);
```

```
00137 void                    EEPROM_SPI_IO_WriteD
ummy(void);
00138
00139 /**
00140   * @}
00141   */
00142
00143 /**
00144   * @}
00145   */
00146
00147 /**
00148   * @}
00149   */
00150
00151 /**
00152   * @}
00153   */
00154
00155 /**
00156   * @}
00157   */
00158
00159 #ifdef __cplusplus
00160 }
00161 #endif
00162
00163 #endif /* __H */
00164
00165 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32303C_EVAL BSP User Manual

stm32303c_eval_eeprom.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32303c_eval_eeprom.c
00004   * @author  MCD Application Team
00005   * @brief   This file provides a set of functions needed to manage a M24LR64
00006   *          or M24M01 I2C EEPROM memory, or a M95M01 SPI EEPROM memory.
00007   *
00008   *          ===================================
00009   *          Notes:
00010   *           - This driver is intended for STM32F30x families devices only.
00011   *           - The I2C EEPROM memory (M24LR64) is available on separate daughter
00012   *             board ANT7-M24LR-A, which is provided with the STM32F303C
00013   *             EVAL board.
00014   *             To use this driver with M24LR64, you have to connect
```

```
00015   *              the ANT7-M24LR-A to CN1 conn
ector of STM32F303C EVAL board.
00016   *              Jumper JP5 and JP6 needs to
be set in I2C2 position.
00017   *              Jumper JP8 (E2P WC) needs to
 be set.
00018   *
00019   *            - The I2C EEPROM memory (M24M0
1) is mounted on the STM32F303C
00020   *              EVAL board.
00021   *              Jumper JP5 and JP6 needs to
be set in I2C2_F position.
00022   *              Jumper JP8 (E2P WC) needs to
 be set.
00023   *
00024   *            - The SPI EEPROM memory (M95M0
1) is available directly
00025   *              on STM32F303C EVAL board.
00026   *          ==============================
=====================================
00027   *
00028   *          It implements a high level comm
unication layer for read and write
00029   *          from/to this memory. The needed
 STM32F30x hardware resources (I2C,
00030   *          SPI and GPIO) are defined in st
m32303c_eval.h file,
00031   *          and the initialization is perfo
rmed depending of EEPROMs
00032   *          in EEPROM_I2C_IO_Init() or EEPR
OM_SPI_IO_Init() functions
00033   *          declared in stm32303c_eval.c fi
le.
00034   *          You can easily tailor this driv
er to any other development board,
00035   *          by just adapting the defines fo
r hardware resources and
```

```
00036   *          EEPROM_I2C_IO_Init() or EEPROM_
SPI_IO_Init() functions.
00037   *
00038   *          @note In this driver, basic rea
d and write functions
00039   *          (BSP_EEPROM_ReadBuffer() and BS
P_EEPROM_WriteBuffer())
00040   *          use Polling mode to perform the
 data transfer to/from EEPROM memories.
00041   *     +-----------------------------------
------------------------------+
00042   *     |                Pin assignment for M
24M01 EEPROM                  |
00043   *     +-----------------------------------
----+-----------+-------------+
00044   *     |  STM32F30x I2C Pins
    |   EEPROM  |   Pin       |
00045   *     +-----------------------------------
----+-----------+-------------+
00046   *     | .
    |   NC      |    1        |
00047   *     | .
    |   E1(VDD) |    2 (3.3V) |
00048   *     | .
    |   E2(GND) |    3  (0V)  |
00049   *     | .
    |   VSS     |    4  (0V)  |
00050   *     | EEPROM_I2C_SDA_PIN/ SDA
    |   SDA     |    5        |
00051   *     | EEPROM_I2C_SCL_PIN/ SCL
    |   SCL     |    6        |
00052   *     | .
    |   /WC(VSS)|    7 (0V)   |
00053   *     | .
    |   VDD     |    8 (3.3V) |
00054   *     +-----------------------------------
----+-----------+-------------+
```

```
00055   *     +-----------------------------------
-------------------------------+
00056   *     |                Pin assignment for M
24LR64 EEPROM                  |
00057   *     +-----------------------------------
----+-----------+-------------+
00058   *     |  STM32F30x I2C Pins
    |   EEPROM  |   Pin       |
00059   *     +-----------------------------------
----+-----------+-------------+
00060   *     | .
    |   E0(GND) |    1  (0V)  |
00061   *     | .
    |   AC0     |    2        |
00062   *     | .
    |   AC1     |    3        |
00063   *     | .
    |   VSS     |    4  (0V)  |
00064   *     | EEPROM_I2C_SDA_PIN/ SDA
    |   SDA     |    5        |
00065   *     | EEPROM_I2C_SCL_PIN/ SCL
    |   SCL     |    6        |
00066   *     | .
    |   E1(GND) |    7  (0V)  |
00067   *     | .
    |   VDD     |    8 (3.3V) |
00068   *     +-----------------------------------
----+-----------+-------------+
00069   *
00070   *     +-----------------------------------
-------------------------------+
00071   *     |                 Pin assignment for
M95M01 EEPROM                  |
00072   *     +-----------------------------------
----+-----------+-------------+
00073   *     |  STM32F30x SPI Pins
    |   EEPROM  |   Pin       |
```

```
00074   *     +-----------------------------------
----+-----------+-------------+
00075   *     | EEPROM_CS_PIN
    | ChipSelect|    1 (/S)   |
00076   *     | EEPROM_SPI_MISO_PIN/ MISO
    |  DataOut  |    2 (Q)    |
00077   *     | EEPROM_SPI_WP_PIN
    | WR Protect|    3 (/W)   |
00078   *     | .
    |   GND     |    4 (0 V)  |
00079   *     | EEPROM_SPI_MOSI_PIN/ MOSI
    |  DataIn   |    5 (D)    |
00080   *     | EEPROM_SPI_SCK_PIN/ SCLK
    |   Clock   |    6 (C)    |
00081   *     | .
    |   Hold    |    7 (/HOLD)|
00082   *     | .
    |   VCC     |    8 (3.3 V)|
00083   *     +-----------------------------------
----+-----------+-------------+
00084   ******************************************
**********************************
00085   * @attention
00086   *
00087   * <h2><center>&copy; COPYRIGHT(c) 2016 STM
icroelectronics</center></h2>
00088   *
00089   * Redistribution and use in source and bin
ary forms, with or without modification,
00090   * are permitted provided that the followin
g conditions are met:
00091   *   1. Redistributions of source code must
 retain the above copyright notice,
00092   *      this list of conditions and the fol
lowing disclaimer.
00093   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
```

```
00094   *      this list of conditions and the fol
lowing disclaimer in the documentation
00095   *      and/or other materials provided wit
h the distribution.
00096   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00097   *      may be used to endorse or promote p
roducts derived from this software
00098   *      without specific prior written perm
ission.
00099   *
00100   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00101   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00102   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00103   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00104   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00105   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
00106   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00107   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00108   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00109   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00110   *
00111   ****************************************
****************************************
00112   */
00113
00114 /* Includes ------------------------------
```

```
----------------------------------*/
00115 #include "stm32303c_eval_eeprom.h"
00116
00117
00118 /** @addtogroup BSP
00119   * @{
00120   */
00121
00122 /** @addtogroup STM32303C_EVAL
00123   * @{
00124   */
00125
00126 /** @addtogroup STM32303C_EVAL_EEPROM
00127   * @brief      This file includes the I2C E
EPROM and SPI EEPROM driver
00128   *             of STM32303C-EVAL board.
00129   * @{
00130   */
00131
00132 /** @addtogroup STM32303C_EVAL_EEPROM_Privat
e_Variables
00133   * @{
00134   */
00135 __IO uint16_t  EEPROMAddress = 0;
00136 __IO uint16_t  EEPROMPageSize = 0;
00137 __IO uint16_t  EEPROMDataRead;
00138 __IO uint8_t   EEPROMDataWrite;
00139
00140 static EEPROM_DrvTypeDef *EEPROM_SelectedDev
ice = 0;
00141 /**
00142   * @}
00143   */
00144
00145 /** @addtogroup STM32303C_EVAL_EEPROM_Privat
e_Functions
00146   * @{
```

```
00147   */
00148 static uint32_t EEPROM_I2C_Init(void);
00149 static uint32_t EEPROM_I2C_ReadBuffer(uint8_
t* pBuffer, uint16_t ReadAddr, uint32_t* NumByteTo
Read);
00150 static uint32_t EEPROM_I2C_WritePage(uint8_t
* pBuffer, uint16_t WriteAddr, uint32_t* NumByteTo
Write);
00151 static uint32_t EEPROM_I2C_WaitEepromStandby
State(void);
00152
00153 static uint32_t EEPROM_SPI_Init(void);
00154 static uint32_t EEPROM_SPI_ReadBuffer(uint8_
t* pBuffer, uint16_t ReadAddr, uint32_t* NumByteTo
Read);
00155 static uint32_t EEPROM_SPI_WritePage(uint8_t
* pBuffer, uint16_t WriteAddr, uint32_t* NumByteTo
Write);
00156 static uint32_t EEPROM_SPI_WaitEepromStandby
State(void);
00157 /**
00158   * @}
00159   */
00160
00161 /** @defgroup STM32303C_EVAL_EEPROM_Private_
Types Private Types
00162   * @{
00163   */
00164
00165 /* EEPROM I2C driver typedef */
00166 EEPROM_DrvTypeDef EEPROM_I2C_Drv =
00167 {
00168   EEPROM_I2C_Init,
00169   EEPROM_I2C_ReadBuffer,
00170   EEPROM_I2C_WritePage
00171 };
00172
```

```
00173 /* EEPROM SPI driver typedef */
00174 EEPROM_DrvTypeDef EEPROM_SPI_Drv =
00175 {
00176   EEPROM_SPI_Init,
00177   EEPROM_SPI_ReadBuffer,
00178   EEPROM_SPI_WritePage
00179 };
00180 /**
00181   * @}
00182   */
00183
00184 /** @addtogroup STM32303C_EVAL_EEPROM_Export
ed_Functions
00185   * @{
00186   */
00187
00188 /**
00189   * @brief  Initializes peripherals used by
the EEPROM device selected.
00190   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00191   *         different from EEPROM_OK (0)
00192   */
00193 uint32_t BSP_EEPROM_Init(void)
00194 {
00195   if(EEPROM_SelectedDevice->Init != 0)
00196   {
00197     return (EEPROM_SelectedDevice->Init());
00198   }
00199   else
00200   {
00201     return EEPROM_FAIL;
00202   }
00203 }
00204
00205 /**
00206   * @brief  Select the EEPROM device to comm
```

```
unicate.
00207   * @param  DeviceID Specifies the EEPROM de
vice to be selected.
00208   *   This parameter can be one of following
 parameters:
00209   *     @arg BSP_EEPROM_M24LR64
00210   *     @arg BSP_EEPROM_M24M01
00211   *     @arg BSP_EEPROM_M95M01
00212   *
00213   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00214   *         different from EEPROM_OK (0)
00215   */
00216 void BSP_EEPROM_SelectDevice(uint8_t DeviceI
D)
00217 {
00218   switch(DeviceID)
00219   {
00220   case BSP_EEPROM_M24LR64 :
00221     EEPROM_SelectedDevice = &EEPROM_I2C_Drv;
00222     break;
00223
00224   case BSP_EEPROM_M24M01 :
00225     EEPROM_SelectedDevice = &EEPROM_I2C_Drv;
00226     break;
00227
00228   case BSP_EEPROM_M95M01 :
00229     EEPROM_SelectedDevice = &EEPROM_SPI_Drv;
00230     break;
00231
00232   default:
00233     break;
00234   }
00235 }
00236
00237 /**
00238   * @brief  Reads a block of data from the E
```

```
EPROM device selected.
00239   * @param  pBuffer pointer to the buffer th
at receives the data read from
00240   *         the EEPROM.
00241   * @param  ReadAddr EEPROM's internal addre
ss to start reading from.
00242   * @param  NumByteToRead pointer to the var
iable holding number of bytes to
00243   *         be read from the EEPROM.
00244   *
00245   *        @note The variable pointed by Num
ByteToRead is reset to 0 when all the
00246   *              data are read from the EEPR
OM. Application should monitor this
00247   *              variable in order know when
 the transfer is complete.
00248   *
00249   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00250   *         different from EEPROM_OK (0) or
the timeout user callback.
00251   */
00252 uint32_t BSP_EEPROM_ReadBuffer(uint8_t* pBuf
fer, uint16_t ReadAddr, uint32_t* NumByteToRead)
00253 {
00254   if(EEPROM_SelectedDevice->ReadBuffer != 0)
00255   {
00256     return (EEPROM_SelectedDevice->ReadBuffer
(pBuffer, ReadAddr, NumByteToRead));
00257   }
00258   else
00259   {
00260     return EEPROM_FAIL;
00261   }
00262 }
00263
00264 /**
```

```
00265   * @brief  Writes buffer of data to the EEP
ROM device selected.
00266   * @param  pBuffer pointer to the buffer  c
ontaining the data to be written
00267   *         to the EEPROM.
00268   * @param  WriteAddr EEPROM's internal addr
ess to write to.
00269   * @param  NumByteToWrite number of bytes t
o write to the EEPROM.
00270   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00271   *         different from EEPROM_OK (0) or
the timeout user callback.
00272   */
00273 uint32_t BSP_EEPROM_WriteBuffer(uint8_t* pBu
ffer, uint16_t WriteAddr, uint32_t NumByteToWrite)
00274 {
00275   uint16_t numofpage = 0, numofsingle = 0, c
ount = 0;
00276   uint16_t addr = 0;
00277   uint32_t dataindex = 0;
00278   uint32_t status = EEPROM_OK;
00279
00280   addr = WriteAddr % EEPROMPageSize;
00281   count = EEPROMPageSize - addr;
00282   numofpage =  NumByteToWrite / EEPROMPageSi
ze;
00283   numofsingle = NumByteToWrite % EEPROMPageS
ize;
00284
00285   if(EEPROM_SelectedDevice->WritePage == 0)
00286   {
00287     return EEPROM_FAIL;
00288   }
00289
00290   /*!< If WriteAddr is EEPROM_PAGESIZE align
ed  */
```

```
00291   if(addr == 0)
00292   {
00293     /*!< If NumByteToWrite < EEPROM_PAGESIZE
 */
00294     if(numofpage == 0)
00295     {
00296       /* Store the number of data to be writ
ten */
00297       dataindex = numofsingle;
00298       /* Start writing data */
00299       status = EEPROM_SelectedDevice->WriteP
age(pBuffer, WriteAddr, (uint32_t*)(&dataindex));
00300       if (status != EEPROM_OK)
00301       {
00302         return status;
00303       }
00304     }
00305     /*!< If NumByteToWrite > EEPROM_PAGESIZE
 */
00306     else
00307     {
00308       while(numofpage--)
00309       {
00310         /* Store the number of data to be wr
itten */
00311         dataindex = EEPROMPageSize;
00312         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00313         if (status != EEPROM_OK)
00314         {
00315           return status;
00316         }
00317
00318         WriteAddr +=  EEPROMPageSize;
00319         pBuffer += EEPROMPageSize;
00320       }
```

```
00321
00322       if(numofsingle!=0)
00323       {
00324         /* Store the number of data to be wr
itten */
00325         dataindex = numofsingle;
00326         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00327         if (status != EEPROM_OK)
00328         {
00329           return status;
00330         }
00331           }
00332         }
00333     }
00334   /*!< If WriteAddr is not EEPROM_PAGESIZE a
ligned  */
00335   else
00336   {
00337     /*!< If NumByteToWrite < EEPROM_PAGESIZE
 */
00338     if(numofpage== 0)
00339     {
00340       /*!< If the number of data to be writt
en is more than the remaining space
00341       in the current page: */
00342       if (NumByteToWrite > count)
00343       {
00344         /* Store the number of data to be wr
itten */
00345         dataindex = count;
00346         /*!< Write the data contained in sam
e page */
00347         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
```

```
00348         if (status != EEPROM_OK)
00349         {
00350           return status;
00351         }
00352 
00353         /* Store the number of data to be wr
itten */
00354         dataindex = (NumByteToWrite - count)
;
00355         /*!< Write the remaining data in the
 following page */
00356         status = EEPROM_SelectedDevice->Writ
ePage((uint8_t*)(pBuffer + count), (WriteAddr + co
unt), (uint32_t*)(&dataindex));
00357         if (status != EEPROM_OK)
00358         {
00359           return status;
00360         }
00361       }
00362       else
00363       {
00364         /* Store the number of data to be wr
itten */
00365         dataindex = numofsingle;
00366         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00367         if (status != EEPROM_OK)
00368         {
00369           return status;
00370         }
00371         }
00372       }
00373     /*!< If NumByteToWrite > EEPROM_PAGESIZE
 */
00374     else
00375     {
```

```
00376       NumByteToWrite -= count;
00377       numofpage =  NumByteToWrite / EEPROMPa
geSize;
00378       numofsingle = NumByteToWrite % EEPROMP
ageSize;
00379
00380       if(count != 0)
00381       {
00382         /* Store the number of data to be wr
itten */
00383         dataindex = count;
00384         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00385         if (status != EEPROM_OK)
00386         {
00387           return status;
00388         }
00389         WriteAddr += count;
00390         pBuffer += count;
00391       }
00392
00393       while(numofpage--)
00394       {
00395         /* Store the number of data to be wr
itten */
00396         dataindex = EEPROMPageSize;

00397         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00398         if (status != EEPROM_OK)
00399         {
00400           return status;
00401         }
00402         WriteAddr +=  EEPROMPageSize;
00403         pBuffer += EEPROMPageSize;
```

```
00404       }
00405       if(numofsingle != 0)
00406       {
00407         /* Store the number of data to be wr
itten */
00408         dataindex = numofsingle;
00409         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00410         if (status != EEPROM_OK)
00411         {
00412           return status;
00413         }
00414       }
00415     }
00416   }
00417
00418   /* If all operations OK, return EEPROM_OK
(0) */
00419   return EEPROM_OK;
00420 }
00421
00422 /**
00423   * @brief  Basic management of the timeout
situation.
00424   * @retval None.
00425   */
00426 __weak void BSP_EEPROM_TIMEOUT_UserCallback(
void)
00427 {
00428 }
00429 /**
00430   * @}
00431   */
00432
00433 /** @addtogroup STM32303C_EVAL_EEPROM_Privat
e_Functions
```

```
00434   * @{
00435   */
00436
00437 /**
00438   * @brief  Initializes peripherals used by
the I2C EEPROM driver.
00439   * @note There are 2 different versions of
M24LR64 (A01 & A02).
00440   *             Then try to connect on 1st o
ne (EEPROM_I2C_ADDRESS_A01)
00441   *             and if problem, check the 2n
d one (EEPROM_I2C_ADDRESS_A02)
00442   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00443   *         different from EEPROM_OK (0)
00444   */
00445 static uint32_t EEPROM_I2C_Init(void)
00446 {
00447   EEPROM_I2C_IO_Init();
00448
00449   /*Select the EEPROM address for M24LR64 A0
2 and check if OK*/
00450   EEPROMAddress = EEPROM_ADDRESS_M24LR64_A01
;
00451   EEPROMPageSize = EEPROM_PAGESIZE_M24LR64;
00452   if (EEPROM_I2C_IO_IsDeviceReady(EEPROMAddr
ess, EEPROM_MAX_TRIALS) != HAL_OK)
00453   {
00454     /*Select the EEPROM address for M24LR64
A01 and check if OK*/
00455     EEPROMAddress = EEPROM_ADDRESS_M24LR64_A
02;
00456     EEPROMPageSize = EEPROM_PAGESIZE_M24LR64
;
00457     if (EEPROM_I2C_IO_IsDeviceReady(EEPROMAd
dress, EEPROM_MAX_TRIALS) != HAL_OK)
00458     {
```

```
00459       /*Select the EEPROM address for M24M01
 and check if OK*/
00460       EEPROMAddress = EEPROM_ADDRESS_M24M01;
00461       EEPROMPageSize = EEPROM_PAGESIZE_M24M01
;
00462       if (EEPROM_I2C_IO_IsDeviceReady(EEPROM
Address, EEPROM_MAX_TRIALS) != HAL_OK)
00463       {
00464         return EEPROM_FAIL;
00465       }
00466     }
00467   }
00468
00469   return EEPROM_OK;
00470 }
00471
00472 /**
00473   * @brief  Reads a block of data from the I
2C EEPROM.
00474   * @param  pBuffer pointer to the buffer th
at receives the data read from
00475   *         the EEPROM.
00476   * @param  ReadAddr EEPROM's internal addre
ss to start reading from.
00477   * @param  NumByteToRead pointer to the var
iable holding number of bytes to
00478   *         be read from the EEPROM.
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_ReadBuffer(uint8_
t* pBuffer, uint16_t ReadAddr, uint32_t* NumByteTo
Read)
00484 {
```

```
00485   uint32_t buffersize = *NumByteToRead;
00486
00487   if (EEPROM_I2C_IO_ReadData(EEPROMAddress,
ReadAddr, pBuffer, buffersize) != HAL_OK)
00488   {
00489     return EEPROM_FAIL;
00490   }
00491
00492   /* If all operations OK, return EEPROM_OK
(0) */
00493   return EEPROM_OK;
00494 }
00495
00496 /**
00497   * @brief  Writes more than one byte to the
 EEPROM with a single WRITE cycle.
00498   *
00499   * @note   The number of bytes (combined to
 write start address) must not
00500   *         cross the EEPROM page boundary.
This function can only write into
00501   *         the boundaries of an EEPROM page.

00502   *         This function doesn't check on b
oundaries condition (in this driver
00503   *         the function BSP_EEPROM_WriteBuf
fer() which calls EEPROM_WritePage() is
00504   *         responsible of checking on Page
boundaries).
00505   *
00506   * @param  pBuffer pointer to the buffer co
ntaining the data to be written to
00507   *         the EEPROM.
00508   * @param  WriteAddr EEPROM's internal addr
ess to write to.
00509   * @param  NumByteToWrite pointer to the va
riable holding number of bytes to
```

```
00510   *          be written into the EEPROM.
00511   *
00512   *        @note The variable pointed by Num
ByteToWrite is reset to 0 when all the
00513   *              data are written to the EEP
ROM. Application should monitor this
00514   *              variable in order know when
 the transfer is complete.
00515   *
00516   *
00517   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00518   *         different from EEPROM_OK (0) or
the timeout user callback.
00519   */
00520 static uint32_t EEPROM_I2C_WritePage(uint8_t
* pBuffer, uint16_t WriteAddr, uint32_t* NumByteTo
Write)
00521 {
00522   uint32_t buffersize = *NumByteToWrite;
00523
00524   if (EEPROM_I2C_IO_WriteData(EEPROMAddress,
 WriteAddr, pBuffer, buffersize) != HAL_OK)
00525   {
00526     return EEPROM_FAIL;
00527   }
00528
00529   /* Wait for EEPROM Standby state */
00530   if (EEPROM_I2C_WaitEepromStandbyState() !=
EEPROM_OK)
00531   {
00532     return EEPROM_FAIL;
00533   }
00534
00535   return EEPROM_OK;
00536 }
00537
```

```
00538 /**
00539   * @brief  Wait for EEPROM I2C Standby stat
e.
00540   *
00541   * @note  This function allows to wait and
check that EEPROM has finished the
00542   *        last operation. It is mostly used
 after Write operation: after receiving
00543   *        the buffer to be written, the EEP
ROM may need additional time to actually
00544   *        perform the write operation. Duri
ng this time, it doesn't answer to
00545   *        I2C packets addressed to it. Once
 the write operation is complete
00546   *        the EEPROM responds to its addres
s.
00547   *
00548   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00549   *         different from EEPROM_OK (0) or
the timeout user callback.
00550   */
00551 static uint32_t EEPROM_I2C_WaitEepromStandby
State(void)
00552 {
00553     /* Check if the maximum allowed number o
f trials has bee reached */
00554   if (EEPROM_I2C_IO_IsDeviceReady(EEPROMAddr
ess, EEPROM_MAX_TRIALS) != HAL_OK)
00555   {
00556     /* If the maximum number of trials has b
een reached, exit the function */
00557     BSP_EEPROM_TIMEOUT_UserCallback();
00558     return EEPROM_TIMEOUT;
00559   }
00560   return EEPROM_OK;
00561 }
```

```
00562
00563 /**
00564   * @brief  Initializes peripherals used by
the SPI EEPROM driver.
00565   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00566   *         different from EEPROM_OK (0)
00567   */
00568 static uint32_t EEPROM_SPI_Init(void)
00569 {
00570   EEPROM_SPI_IO_Init();
00571
00572   /* Check if EEPROM SPI is ready to use */
00573   EEPROMPageSize = EEPROM_PAGESIZE_M95M01;
00574   if(EEPROM_SPI_WaitEepromStandbyState() !=
HAL_OK)
00575   {
00576     return EEPROM_FAIL;
00577   }
00578   return EEPROM_OK;
00579 }
00580
00581 /**
00582   * @brief  Reads a block of data from the S
PI EEPROM.
00583   * @param  pBuffer pointer to the buffer th
at receives the data read from
00584   *         the EEPROM.
00585   * @param  ReadAddr EEPROM's internal addre
ss to start reading from.
00586   * @param  NumByteToRead pointer to the var
iable holding number of bytes to
00587   *         be read from the EEPROM.
00588   *
00589   *        @note The variable pointed by Num
ByteToRead is reset to 0 when all the
00590   *              data are read from the EEPR
```

```
OM. Application should monitor this
00591   *               variable in order know when
 the transfer is complete.
00592   *
00593   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00594   *         different from EEPROM_OK (0) or
the timeout user callback.
00595   */
00596 static uint32_t EEPROM_SPI_ReadBuffer(uint8_
t* pBuffer, uint16_t ReadAddr, uint32_t* NumByteTo
Read)
00597 {
00598   uint32_t buffersize = *NumByteToRead;
00599   EEPROM_SPI_IO_ReadData(ReadAddr, pBuffer,
buffersize);
00600
00601     /* Wait for EEPROM Standby state */
00602   if (EEPROM_SPI_WaitEepromStandbyState() !=
EEPROM_OK)
00603   {
00604     return EEPROM_FAIL;
00605   }
00606
00607   return EEPROM_OK;
00608 }
00609
00610 /**
00611   * @brief  Writes more than one byte to the
 EEPROM with a single WRITE cycle.
00612   *
00613   * @note   The number of bytes (combined to
 write start address) must not
00614   *         cross the EEPROM page boundary.
This function can only write into
00615   *         the boundaries of an EEPROM page.
```

```
00616   *         This function doesn't check on b
oundaries condition (in this driver
00617   *         the function BSP_EEPROM_WriteBuf
fer() which calls EEPROM_WritePage() is
00618   *         responsible of checking on Page
boundaries).
00619   *
00620   * @param  pBuffer pointer to the buffer co
ntaining the data to be written to
00621   *         the EEPROM.
00622   * @param  WriteAddr EEPROM's internal addr
ess to write to.
00623   * @param  NumByteToWrite pointer to the va
riable holding number of bytes to
00624   *         be written into the EEPROM.
00625   *
00626   *        @note The variable pointed by Num
ByteToWrite is reset to 0 when all the
00627   *              data are written to the EEP
ROM. Application should monitor this
00628   *              variable in order know when
 the transfer is complete.
00629   *
00630   *
00631   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00632   *         different from EEPROM_OK (0) or
the timeout user callback.
00633   */
00634 static uint32_t EEPROM_SPI_WritePage(uint8_t
* pBuffer, uint16_t WriteAddr, uint32_t* NumByteTo
Write)
00635 {
00636   uint32_t buffersize = *NumByteToWrite;
00637
00638   if (EEPROM_SPI_IO_WriteData(WriteAddr, pBu
ffer, buffersize) !=  HAL_OK)
```

```
00639   {
00640     return EEPROM_FAIL;
00641   }
00642
00643   /* Wait for EEPROM Standby state */
00644   if (EEPROM_SPI_WaitEepromStandbyState() !=
EEPROM_OK)
00645   {
00646     return EEPROM_FAIL;
00647   }
00648
00649   return EEPROM_OK;
00650 }
00651
00652 /**
00653   * @brief  Wait for EEPROM SPI Standby stat
e.
00654   *
00655   * @note  This function allows to wait and
check that EEPROM has finished the
00656   *        last operation. It is mostly used
 after Write operation.
00657   *
00658   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00659   *         different from EEPROM_OK (0) or
the timeout user callback.
00660   */
00661 static uint32_t EEPROM_SPI_WaitEepromStandby
State(void)
00662 {
00663   if(EEPROM_SPI_IO_WaitEepromStandbyState()
!= HAL_OK)
00664   {
00665     BSP_EEPROM_TIMEOUT_UserCallback();
00666     return EEPROM_TIMEOUT;
00667   }
```

```
00668   return EEPROM_OK;
00669 }
00670
00671 /**
00672   * @}
00673   */
00674
00675 /**
00676   * @}
00677   */
00678
00679 /**
00680   * @}
00681   */
00682
00683 /**
00684   * @}
00685   */
00686
00687 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32303C_EVAL BSP User Manual

Exported Types

STM32303C-EVAL LCD

Data Structures

struct	**LCD_DrawPropTypeDef**

STM32303C_EVAL BSP User Manual

stm32303c_eval_lcd.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32303c_eval_lcd.h
00004   * @author  MCD Application Team
00005   * @brief   This file contains all the functions prototypes for the
00006   *          stm32303c_eval_lcd.c driver.
00007   ******************************************************************************
00008   * @attention
00009   *
00010   * <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>
00011   *
00012   * Redistribution and use in source and binary forms, with or without modification,
00013   * are permitted provided that the following conditions are met:
00014   *   1. Redistributions of source code must retain the above copyright notice,
00015   *      this list of conditions and the fol
```

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

```
00035   */
00036
00037 /* Define to prevent recursive inclusion ---
-------------------------------------*/
00038 #ifndef __STM32303C_EVAL_LCD_H
00039 #define __STM32303C_EVAL_LCD_H
00040
00041 #ifdef __cplusplus
00042  extern "C" {
00043 #endif
00044
00045 /* Includes ------------------------------
-------------------------------------*/
00046 #include "stm32303c_eval.h"
00047 #include "../Components/hx8347d/hx8347d.h"

00048 #include "../Components/spfd5408/spfd5408.h"

00049 #include "../../../Utilities/Fonts/fonts.h"

00050 /** @addtogroup BSP
00051   * @{
00052   */
00053
00054 /** @addtogroup STM32303C_EVAL
00055   * @{
00056   */
00057
00058 /** @defgroup STM32303C_EVAL_LCD STM32303C-E
VAL LCD
00059   * @{
00060   */
00061
00062 /** @defgroup STM32303C_EVAL_LCD_Private_Def
ines Private Defines
00063   * @{
00064   */
```

```
00065 /**
00066   * @}
00067   */
00068
00069 /** @defgroup STM32303C_EVAL_LCD_Private_Mac
ros Private Macros
00070   * @{
00071   */
00072 /**
00073   * @}
00074   */
00075
00076 /** @defgroup STM32303C_EVAL_LCD_Private_Var
iables Private Variables
00077   * @{
00078   */
00079 /**
00080   * @}
00081   */
00082
00083 /** @defgroup STM32303C_EVAL_LCD_Private_Fun
ctions Private Functions
00084   * @{
00085   */
00086 /**
00087   * @}
00088   */
00089
00090 /** @defgroup STM32303C_EVAL_LCD_Exported_Ty
pes Exported Types
00091   * @{
00092   */
00093 typedef struct
00094 {
00095   uint32_t TextColor;
00096   uint32_t BackColor;
00097   sFONT    *pFont;
```

```
00098
00099 }LCD_DrawPropTypeDef;
00100 /**
00101   * @}
00102   */
00103
00104 /** @defgroup STM32303C_EVAL_LCD_Exported_Co
nstants Exported Constants
00105   * @{
00106   */
00107 /**
00108   * @brief  LCD status structure definition

00109   */
00110 #define   LCD_OK         0x00
00111 #define   LCD_ERROR      0x01
00112 #define   LCD_TIMEOUT    0x02
00113
00114 typedef struct
00115 {
00116   int16_t X;
00117   int16_t Y;
00118
00119 }Point, * pPoint;
00120
00121 /**
00122   * @brief  Line mode structures definition

00123   */
00124 typedef enum
00125 {
00126   CENTER_MODE             = 0x01,    /*!< Ce
nter mode */
00127   RIGHT_MODE              = 0x02,    /*!< Ri
ght mode  */
00128   LEFT_MODE               = 0x03     /*!< Le
ft mode   */
```

```
00129
00130 }Line_ModeTypdef;
00131
00132 /**
00133   * @brief  LCD color
00134   */
00135 #define LCD_COLOR_BLUE          0x001F
00136 #define LCD_COLOR_GREEN         0x07E0
00137 #define LCD_COLOR_RED           0xF800
00138 #define LCD_COLOR_CYAN          0x07FF
00139 #define LCD_COLOR_MAGENTA       0xF81F
00140 #define LCD_COLOR_YELLOW        0xFFE0
00141 #define LCD_COLOR_LIGHTBLUE     0x841F
00142 #define LCD_COLOR_LIGHTGREEN    0x87F0
00143 #define LCD_COLOR_LIGHTRED      0xFC10
00144 #define LCD_COLOR_LIGHTCYAN     0x87FF
00145 #define LCD_COLOR_LIGHTMAGENTA  0xFC1F
00146 #define LCD_COLOR_LIGHTYELLOW   0xFFF0
00147 #define LCD_COLOR_DARKBLUE      0x0010
00148 #define LCD_COLOR_DARKGREEN     0x0400
00149 #define LCD_COLOR_DARKRED       0x8000
00150 #define LCD_COLOR_DARKCYAN      0x0410
00151 #define LCD_COLOR_DARKMAGENTA   0x8010
00152 #define LCD_COLOR_DARKYELLOW    0x8400
00153 #define LCD_COLOR_WHITE         0xFFFF
00154 #define LCD_COLOR_LIGHTGRAY     0xD69A
00155 #define LCD_COLOR_GRAY          0x8410
00156 #define LCD_COLOR_DARKGRAY      0x4208
00157 #define LCD_COLOR_BLACK         0x0000
00158 #define LCD_COLOR_BROWN         0xA145
00159 #define LCD_COLOR_ORANGE        0xFD20
00160
00161 /**
00162   * @brief LCD default font
00163   */
00164 #define LCD_DEFAULT_FONT         Font24
```

```
00165
00166 /**
00167   * @}
00168   */
00169
00170 /** @defgroup STM32303C_EVAL_LCD_Exported_Fu
nctions Exported Functions
00171   * @{
00172   */
00173 uint8_t  BSP_LCD_Init(void);
00174 uint32_t BSP_LCD_GetXSize(void);
00175 uint32_t BSP_LCD_GetYSize(void);
00176
00177 uint16_t BSP_LCD_GetTextColor(void);
00178 uint16_t BSP_LCD_GetBackColor(void);
00179 void     BSP_LCD_SetTextColor(__IO uint16_t
Color);
00180 void     BSP_LCD_SetBackColor(__IO uint16_t
Color);
00181 void     BSP_LCD_SetFont(sFONT *pFonts);
00182 sFONT    *BSP_LCD_GetFont(void);
00183
00184 void     BSP_LCD_Clear(uint16_t Color);
00185 void     BSP_LCD_ClearStringLine(uint16_t Li
ne);
00186 void     BSP_LCD_DisplayStringAtLine(uint16_
t Line, uint8_t *pText);
00187 void     BSP_LCD_DisplayStringAt(uint16_t Xp
os, uint16_t Ypos, uint8_t *pText, Line_ModeTypdef
 Mode);
00188 void     BSP_LCD_DisplayChar(uint16_t Xpos,
uint16_t Ypos, uint8_t Ascii);
00189
00190 uint16_t BSP_LCD_ReadPixel(uint16_t Xpos, ui
nt16_t Ypos);
00191 void     BSP_LCD_DrawHLine(uint16_t Xpos, ui
nt16_t Ypos, uint16_t Length);
```

```
00192 void     BSP_LCD_DrawVLine(uint16_t Xpos, ui
nt16_t Ypos, uint16_t Length);
00193 void     BSP_LCD_DrawLine(uint16_t X1, uint1
6_t Y1, uint16_t X2, uint16_t Y2);
00194 void     BSP_LCD_DrawRect(uint16_t Xpos, uin
t16_t Ypos, uint16_t Width, uint16_t Height);
00195 void     BSP_LCD_DrawCircle(uint16_t Xpos, u
int16_t Ypos, uint16_t Radius);
00196 void     BSP_LCD_DrawPolygon(pPoint pPoints,
 uint16_t PointCount);
00197 void     BSP_LCD_DrawEllipse(int Xpos, int Y
pos, int XRadius, int YRadius);
00198 void     BSP_LCD_DrawBitmap(uint16_t Xpos, u
int16_t Ypos, uint8_t *pBmp);
00199
00200 void     BSP_LCD_FillRect(uint16_t Xpos, uin
t16_t Ypos, uint16_t Width, uint16_t Height);
00201 void     BSP_LCD_FillCircle(uint16_t Xpos, u
int16_t Ypos, uint16_t Radius);
00202 void     BSP_LCD_FillEllipse(int Xpos, int Y
pos, int XRadius, int YRadius);
00203
00204 void     BSP_LCD_DisplayOff(void);
00205 void     BSP_LCD_DisplayOn(void);
00206
00207 /**
00208   * @}
00209   */
00210
00211 /**
00212   * @}
00213   */
00214
00215 /**
00216   * @}
00217   */
00218
```

```
00219 /**
00220   * @}
00221   */
00222
00223 #ifdef __cplusplus
00224 }
00225 #endif
00226
00227 #endif /* __STM32303C_EVAL_LCD_H */
00228
00229 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32303C_EVAL BSP User Manual

stm32303c_eval_lcd.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32303c_eval_lcd.c
00004   * @author  MCD Application Team
00005   * @brief   This file includes the driver for Liquid Crystal Display modules
00006   *          mounted on STM32303C-EVAL evaluation board.
00007   ******************************************************************************
00008   * @attention
00009   *
00010   * <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>
00011   *
00012   * Redistribution and use in source and binary forms, with or without modification,
00013   * are permitted provided that the following conditions are met:
00014   *   1. Redistributions of source code must retain the above copyright notice,
```

```
00015   *      this list of conditions and the fol
lowing disclaimer.
00016   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00017   *      this list of conditions and the fol
lowing disclaimer in the documentation
00018   *      and/or other materials provided wit
h the distribution.
00019   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00020   *      may be used to endorse or promote p
roducts derived from this software
00021   *      without specific prior written perm
ission.
00022   *
00023   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00024   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00025   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00026   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00027   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00028   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
00029   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00030   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00031   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00032   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00033   *
00034   ****************************************
```

```
************************************
00035   */
00036
00037 /* File Info : ------------------------------
-------------------------------------
00038                                       User NOTES

00039 1. How To use this driver:
00040 --------------------------
00041    - This driver is used to drive indirectly
 an LCD TFT.
00042    - This driver supports the AM-240320L8TNQ
W00H (ILI9320),
00043      AM-240320LDTNQW00H (SPFD5408B) and AM24
0320LGTNQW00H (HX8347D) LCD
00044      mounted on MB895 daughter board
00045    - The ILI9320, SPFD5408B and HX8347D comp
onents driver MUST be included with this driver.
00046
00047 2. Driver description:
00048 ---------------------
00049   + Initialization steps:
00050      o Initialize the LCD using the LCD_Init
() function.
00051
00052   + Display on LCD
00053      o Clear the hole LCD using yhe LCD_Clea
r() function or only one specified
00054        string line using the LCD_ClearString
Line() function.
00055      o Display a character on the specified
line and column using the LCD_DisplayChar()
00056        function or a complete string line us
ing the LCD_DisplayStringAtLine() function.
00057      o Display a string line on the specifie
d position (x,y in pixel) and align mode
00058        using the LCD_DisplayStringAtLine() f
```

```
unction.
00059      o Draw and fill a basic shapes (dot, li
ne, rectangle, circle, ellipse, .. bitmap)
00060        on LCD using a set of functions.
00061
00062 ------------------------------------------
-----------------------------------*/
00063
00064 /* Includes ------------------------------
-----------------------------------*/
00065 #include "stm32303c_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 STM32303C_EVAL
00078   * @{
00079   */
00080
00081 /** @addtogroup STM32303C_EVAL_LCD
00082   * @{
00083   */
00084
00085 /** @addtogroup STM32303C_EVAL_LCD_Private_D
efines
00086   * @{
00087   */
00088 #define POLY_X(Z)              ((int32_t)((
pPoints + (Z))->X))
00089 #define POLY_Y(Z)              ((int32_t)((
```

```
pPoints + (Z))->Y))
00090
00091 #define MAX_HEIGHT_FONT         17
00092 #define MAX_WIDTH_FONT          24
00093 #define OFFSET_BITMAP           54
00094 /**
00095   * @}
00096   */
00097
00098 /** @addtogroup STM32303C_EVAL_LCD_Private_M
acros
00099   * @{
00100   */
00101 #define ABS(X)  ((X) > 0 ? (X) : -(X))
00102
00103 /**
00104   * @}
00105   */
00106
00107 /** @addtogroup STM32303C_EVAL_LCD_Private_V
ariables STM32303C_EVAL_LCD_Private_Variables
00108   * @{
00109   */
00110 LCD_DrawPropTypeDef DrawProp;
00111
00112 static LCD_DrvTypeDef  *lcd_drv;
00113
00114 /* Max size of bitmap will based on a font24
 (17x24) */
00115 static uint8_t bitmap[MAX_HEIGHT_FONT*MAX_WI
DTH_FONT*2+OFFSET_BITMAP] = {0};
00116
00117 /**
00118   * @}
00119   */
00120
00121 /** @addtogroup STM32303C_EVAL_LCD_Private_F
```

```
unctions
00122   * @{
00123   */
00124 static void LCD_DrawPixel(uint16_t Xpos, uin
t16_t Ypos, uint16_t RGBCode);
00125 static void LCD_DrawChar(uint16_t Xpos, uint
16_t Ypos, const uint8_t *pChar);
00126 static void LCD_SetDisplayWindow(uint16_t Xp
os, uint16_t Ypos, uint16_t Width, uint16_t Height
);
00127 /**
00128   * @}
00129   */
00130
00131 /** @addtogroup STM32303C_EVAL_LCD_Exported_
Functions
00132   * @{
00133   */
00134
00135 /**
00136   * @brief  Initializes the LCD.
00137   * @retval LCD state
00138   */
00139 uint8_t BSP_LCD_Init(void)
00140 {
00141   uint8_t ret = LCD_ERROR;
00142
00143   /* Default value for draw propriety */
00144   DrawProp.BackColor = 0xFFFF;
00145   DrawProp.pFont     = &Font24;
00146   DrawProp.TextColor = 0x0000;
00147
00148   if(spfd5408_drv.ReadID() == SPFD5408_ID)
00149   {
00150     lcd_drv = &spfd5408_drv;
00151     ret = LCD_OK;
00152   }
```

```
00153   else
00154   {
00155     /*HX8347D_ID connected*/
00156   lcd_drv = &hx8347d_drv;
00157   ret = LCD_OK;
00158   }
00159
00160   if(ret != LCD_ERROR)
00161   {
00162     /* LCD Init */
00163     lcd_drv->Init();
00164
00165     /* Initialize the font */
00166     BSP_LCD_SetFont(&LCD_DEFAULT_FONT);
00167   }
00168
00169   return ret;
00170 }
00171
00172 /**
00173   * @brief  Gets the LCD X size.
00174   * @retval Used LCD X size
00175   */
00176 uint32_t BSP_LCD_GetXSize(void)
00177 {
00178   return(lcd_drv->GetLcdPixelWidth());
00179 }
00180
00181 /**
00182   * @brief  Gets the LCD Y size.
00183   * @retval Used LCD Y size
00184   */
00185 uint32_t BSP_LCD_GetYSize(void)
00186 {
00187   return(lcd_drv->GetLcdPixelHeight());
00188 }
00189
```

```
00190 /**
00191   * @brief  Gets the LCD text color.
00192   * @retval Used text color.
00193   */
00194 uint16_t BSP_LCD_GetTextColor(void)
00195 {
00196   return DrawProp.TextColor;
00197 }
00198
00199 /**
00200   * @brief  Gets the LCD background color.
00201   * @retval Used background color
00202   */
00203 uint16_t BSP_LCD_GetBackColor(void)
00204 {
00205   return DrawProp.BackColor;
00206 }
00207
00208 /**
00209   * @brief  Sets the LCD text color.
00210   * @param  Color Text color code RGB(5-6-5)
00211   * @retval None
00212   */
00213 void BSP_LCD_SetTextColor(uint16_t Color)
00214 {
00215   DrawProp.TextColor = Color;
00216 }
00217
00218 /**
00219   * @brief  Sets the LCD background color.
00220   * @param  Color Background color code RGB(
5-6-5)
00221   * @retval None
00222   */
00223 void BSP_LCD_SetBackColor(uint16_t Color)
00224 {
00225   DrawProp.BackColor = Color;
```

```
00226 }
00227
00228 /**
00229   * @brief  Sets the LCD text font.
00230   * @param  pFonts Font to be used
00231   * @retval None
00232   */
00233 void BSP_LCD_SetFont(sFONT *pFonts)
00234 {
00235   DrawProp.pFont = pFonts;
00236 }
00237
00238 /**
00239   * @brief  Gets the LCD text font.
00240   * @retval Used font
00241   */
00242 sFONT *BSP_LCD_GetFont(void)
00243 {
00244   return DrawProp.pFont;
00245 }
00246
00247 /**
00248   * @brief  Clears the hole LCD.
00249   * @param  Color Color of the background
00250   * @retval None
00251   */
00252 void BSP_LCD_Clear(uint16_t Color)
00253 {
00254   uint32_t counter = 0;
00255
00256   uint32_t color_backup = DrawProp.TextColor
;
00257   DrawProp.TextColor = Color;
00258
00259   for(counter = 0; counter < BSP_LCD_GetYSize
(); counter++)
00260   {
```

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

00280   DrawProp.TextColor = DrawProp.BackColor;;
00281
00282   /* Draw a rectangle with background color
*/
00283   BSP_LCD_FillRect(0, (Line * DrawProp.pFont
->Height), BSP_LCD_GetXSize(), DrawProp.pFont->Hei
ght);
00284
00285   DrawProp.TextColor = colorbackup;
00286   BSP_LCD_SetTextColor(DrawProp.TextColor);
00287 }
00288
```

```
00289 /**
00290   * @brief  Displays one character.
00291   * @param  Xpos Start column address
00292   * @param  Ypos Line where to display the c
haracter shape.
00293   * @param  Ascii Character ascii code
00294   *           This parameter must be a numbe
r between Min_Data = 0x20 and Max_Data = 0x7E
00295   * @retval None
00296   */
00297 void BSP_LCD_DisplayChar(uint16_t Xpos, uint
16_t Ypos, uint8_t Ascii)
00298 {
00299   LCD_DrawChar(Ypos, Xpos, &DrawProp.pFont->
table[(Ascii-' ') *\
00300     DrawProp.pFont->Height * ((DrawProp.pFont
->Width + 7) / 8)]);
00301 }
00302
00303 /**
00304   * @brief  Displays characters on the LCD.
00305   * @param  Xpos X position (in pixel)
00306   * @param  Ypos Y position (in pixel)
00307   * @param  pText Pointer to string to displ
ay on LCD
00308   * @param  Mode Display mode
00309   *          This parameter can be one of th
e following values:
00310   *            @arg  CENTER_MODE
00311   *            @arg  RIGHT_MODE
00312   *            @arg  LEFT_MODE
00313   * @retval None
00314   */
00315 void BSP_LCD_DisplayStringAt(uint16_t Xpos,
uint16_t Ypos, uint8_t *pText, Line_ModeTypdef Mod
e)
00316 {
```

```
00317   uint16_t refcolumn = 1, counter = 0;
00318   uint32_t size = 0, ysize = 0;
00319   uint8_t  *ptr = pText;
00320
00321   /* Get the text size */
00322   while (*ptr++) size ++ ;
00323
00324   /* Characters number per line */
00325   ysize = (BSP_LCD_GetXSize()/DrawProp.pFont
->Width);
00326
00327   switch (Mode)
00328   {
00329   case CENTER_MODE:
00330     {
00331       refcolumn = Xpos + ((ysize - size)* Dr
awProp.pFont->Width) / 2;
00332       break;
00333     }
00334   case LEFT_MODE:
00335     {
00336       refcolumn = Xpos;
00337       break;
00338     }
00339   case RIGHT_MODE:
00340     {
00341       refcolumn = Xpos + ((ysize - size)*Dra
wProp.pFont->Width);
00342       break;
00343     }
00344   default:
00345     {
00346       refcolumn = Xpos;
00347       break;
00348     }
00349   }
00350
```

```
00351   /* Send the string character by character
on lCD */
00352   while ((*pText != 0) & (((BSP_LCD_GetXSize
() - (counter*DrawProp.pFont->Width)) & 0xFFFF) >=
 DrawProp.pFont->Width))
00353   {
00354     /* Display one character on LCD */
00355     BSP_LCD_DisplayChar(refcolumn, Ypos, *pT
ext);
00356     /* Decrement the column position by 16 */

00357     refcolumn += DrawProp.pFont->Width;
00358     /* Point on the next character */
00359     pText++;
00360     counter++;
00361   }
00362 }
00363
00364 /**
00365   * @brief  Displays a character on the LCD.
00366   * @param  Line Line where to display the c
haracter shape
00367   *          This parameter can be one of th
e following values:
00368   *            @arg  0..9: if the Current fo
nts is Font16x24
00369   *            @arg  0..19: if the Current f
onts is Font12x12 or Font8x12
00370   *            @arg  0..29: if the Current f
onts is Font8x8
00371   * @param  pText Pointer to string to displ
ay on LCD
00372   * @retval None
00373   */
00374 void BSP_LCD_DisplayStringAtLine(uint16_t Li
ne, uint8_t *pText)
00375 {
```

```
00376   BSP_LCD_DisplayStringAt(0, LINE(Line),pTex
t, LEFT_MODE);
00377 }
00378
00379 /**
00380   * @brief  Reads an LCD pixel.
00381   * @param  Xpos X position
00382   * @param  Ypos Y position
00383   * @retval RGB pixel color
00384   */
00385 uint16_t BSP_LCD_ReadPixel(uint16_t Xpos, ui
nt16_t Ypos)
00386 {
00387   uint16_t ret = 0;
00388
00389   if(lcd_drv->ReadPixel != NULL)
00390   {
00391     ret = lcd_drv->ReadPixel(Xpos, Ypos);
00392   }
00393
00394   return ret;
00395 }
00396
00397 /**
00398   * @brief  Draws an horizontal line.
00399   * @param  Xpos X position
00400   * @param  Ypos Y position
00401   * @param  Length Line length
00402   * @retval None
00403   */
00404 void BSP_LCD_DrawHLine(uint16_t Xpos, uint16
_t Ypos, uint16_t Length)
00405 {
00406   uint32_t index = 0;
00407
00408   if(lcd_drv->DrawHLine != NULL)
00409   {
```

```
00410     lcd_drv->DrawHLine(DrawProp.TextColor, Y
pos, Xpos, Length);
00411   }
00412   else
00413   {
00414     for(index = 0; index < Length; index++)
00415     {
00416       LCD_DrawPixel((Ypos + index), Xpos, Dr
awProp.TextColor);
00417     }
00418   }
00419 }
00420
00421 /**
00422   * @brief  Draws a vertical line.
00423   * @param  Xpos X position
00424   * @param  Ypos Y position
00425   * @param  Length Line length
00426   * @retval None
00427   */
00428 void BSP_LCD_DrawVLine(uint16_t Xpos, uint16
_t Ypos, uint16_t Length)
00429 {
00430   uint32_t index = 0;
00431
00432   if(lcd_drv->DrawVLine != NULL)
00433   {
00434     LCD_SetDisplayWindow(Ypos, Xpos, 1, Leng
th);
00435     lcd_drv->DrawVLine(DrawProp.TextColor, Y
pos, Xpos, Length);
00436     LCD_SetDisplayWindow(0, 0, BSP_LCD_GetXS
ize(), BSP_LCD_GetYSize());
00437   }
00438   else
00439   {
00440     for(index = 0; index < Length; index++)
```

```
00441     {
00442       LCD_DrawPixel(Ypos, Xpos + index, Draw
Prop.TextColor);
00443     }
00444   }
00445 }
00446
00447 /**
00448   * @brief  Draws an uni-line (between two p
oints).
00449   * @param  X1 Point 1 X position
00450   * @param  Y1 Point 1 Y position
00451   * @param  X2 Point 2 X position
00452   * @param  Y2 Point 2 Y position
00453   * @retval None
00454   */
00455 void BSP_LCD_DrawLine(uint16_t X1, uint16_t
Y1, uint16_t X2, uint16_t Y2)
00456 {
00457   int16_t deltax = 0, deltay = 0, x = 0, y =
 0, xinc1 = 0, xinc2 = 0,
00458   yinc1 = 0, yinc2 = 0, den = 0, num = 0, nu
madd = 0, numpixels = 0,
00459   curpixel = 0;
00460
00461   deltax = ABS(X2 - Y1);         /* The diffe
rence between the x's */
00462   deltay = ABS(X2 - X1);         /* The diffe
rence between the y's */
00463   x = Y1;                        /* Start x o
ff at the first pixel */
00464   y = X1;                        /* Start y o
ff at the first pixel */
00465
00466   if (Y2 >= Y1)                  /* The x-val
ues are increasing */
00467   {
```

```
00468     xinc1 = 1;
00469     xinc2 = 1;
00470   }
00471   else                          /* The x-val
ues are decreasing */
00472   {
00473     xinc1 = -1;
00474     xinc2 = -1;
00475   }
00476
00477   if (X2 >= X1)                 /* The y-val
ues are increasing */
00478   {
00479     yinc1 = 1;
00480     yinc2 = 1;
00481   }
00482   else                          /* The y-val
ues are decreasing */
00483   {
00484     yinc1 = -1;
00485     yinc2 = -1;
00486   }
00487
00488   if (deltax >= deltay)         /* There is
at least one x-value for every y-value */
00489   {
00490     xinc1 = 0;                  /* Don't cha
nge the x when numerator >= denominator */
00491     yinc2 = 0;                  /* Don't cha
nge the y for every iteration */
00492     den = deltax;
00493     num = deltax / 2;
00494     numadd = deltay;
00495     numpixels = deltax;         /* There are
 more x-values than y-values */
00496   }
00497   else                          /* There is
```

```
at least one y-value for every x-value */
00498   {
00499     xinc2 = 0;                  /* Don't cha
nge the x for every iteration */
00500     yinc1 = 0;                  /* Don't cha
nge the y when numerator >= denominator */
00501     den = deltay;
00502     num = deltay / 2;
00503     numadd = deltax;
00504     numpixels = deltay;         /* There are
 more y-values than x-values */
00505   }
00506
00507   for (curpixel = 0; curpixel <= numpixels;
curpixel++)
00508   {
00509     LCD_DrawPixel(x, y, DrawProp.TextColor);
  /* Draw the current pixel */
00510     num += numadd;
  /* Increase the numerator by the top of the frac
tion */
00511     if (num >= den)
  /* Check if numerator >= denominator */
00512     {
00513       num -= den;
  /* Calculate the new numerator value */
00514       x += xinc1;
  /* Change the x as appropriate */
00515       y += yinc1;
  /* Change the y as appropriate */
00516     }
00517     x += xinc2;
  /* Change the x as appropriate */
00518     y += yinc2;
  /* Change the y as appropriate */
00519   }
00520 }
```

```
00521
00522 /**
00523   * @brief  Draws a rectangle.
00524   * @param  Xpos X position
00525   * @param  Ypos Y position
00526   * @param  Width Rectangle width
00527   * @param  Height Rectangle height
00528   * @retval None
00529   */
00530 void BSP_LCD_DrawRect(uint16_t Xpos, uint16_
t Ypos, uint16_t Width, uint16_t Height)
00531 {
00532   /* Draw horizontal lines */
00533   BSP_LCD_DrawHLine(Xpos, Ypos, Width);
00534   BSP_LCD_DrawHLine(Xpos, (Ypos+ Height), Wi
dth);
00535
00536   /* Draw vertical lines */
00537   BSP_LCD_DrawVLine(Xpos, Ypos, Height);
00538   BSP_LCD_DrawVLine((Xpos + Width), Ypos, He
ight);
00539 }
00540
00541 /**
00542   * @brief  Draws a circle.
00543   * @param  Xpos X position
00544   * @param  Ypos Y position
00545   * @param  Radius Circle radius
00546   * @retval None
00547   */
00548 void BSP_LCD_DrawCircle(uint16_t Xpos, uint1
6_t Ypos, uint16_t Radius)
00549 {
00550   int32_t  decision;       /* Decision Varia
ble */
00551   uint32_t  curx;   /* Current X Value */
00552   uint32_t  cury;   /* Current Y Value */
```

```
00553
00554   decision = 3 - (Radius << 1);
00555   curx = 0;
00556   cury = Radius;
00557
00558   while (curx <= cury)
00559   {
00560     LCD_DrawPixel((Ypos + curx), (Xpos - cur
y), DrawProp.TextColor);
00561
00562     LCD_DrawPixel((Ypos - curx), (Xpos - cur
y), DrawProp.TextColor);
00563
00564     LCD_DrawPixel((Ypos + cury), (Xpos - cur
x), DrawProp.TextColor);
00565
00566     LCD_DrawPixel((Ypos - cury), (Xpos - cur
x), DrawProp.TextColor);
00567
00568     LCD_DrawPixel((Ypos + curx), (Xpos + cur
y), DrawProp.TextColor);
00569
00570     LCD_DrawPixel((Ypos - curx), (Xpos + cur
y), DrawProp.TextColor);
00571
00572     LCD_DrawPixel((Ypos + cury), (Xpos + cur
x), DrawProp.TextColor);
00573
00574     LCD_DrawPixel((Ypos - cury), (Xpos + cur
x), DrawProp.TextColor);
00575
00576     /* Initialize the font */
00577     BSP_LCD_SetFont(&LCD_DEFAULT_FONT);
00578
00579     if (decision < 0)
00580     {
00581       decision += (curx << 2) + 6;
```

```
00582     }
00583     else
00584     {
00585       decision += ((curx - cury) << 2) + 10;
00586       cury--;
00587     }
00588     curx++;
00589   }
00590 }
00591
00592 /**
00593   * @brief  Draws an poly-line (between many
 points).
00594   * @param  pPoints Pointer to the points ar
ray
00595   * @param  PointCount Number of points
00596   * @retval None
00597   */
00598 void BSP_LCD_DrawPolygon(pPoint pPoints, uin
t16_t PointCount)
00599 {
00600   int16_t x = 0, y = 0;
00601
00602   if(PointCount < 2)
00603   {
00604     return;
00605   }
00606
00607   BSP_LCD_DrawLine(pPoints->X, pPoints->Y, (
pPoints+PointCount-1)->X, (pPoints+PointCount-1)->
Y);
00608
00609   while(--PointCount)
00610   {
00611     x = pPoints->X;
00612     y = pPoints->Y;
00613     pPoints++;
```

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

```
00644     if (e2 <= x) {
00645       err += ++x*2+1;
00646       if (-y == x && e2 <= y) e2 = 0;
00647     }
00648     if (e2 > y) err += ++y*2+1;
00649   }
00650   while (y <= 0);
00651 }
00652
00653 /**
00654   * @brief  Draws a bitmap picture loaded in
 the internal Flash (32 bpp).
00655   * @param  Xpos Bmp X position in the LCD
00656   * @param  Ypos Bmp Y position in the LCD
00657   * @param  pBmp Pointer to Bmp picture addr
ess in the internal Flash
00658   * @retval None
00659   */
00660 void BSP_LCD_DrawBitmap(uint16_t Xpos, uint1
6_t Ypos, uint8_t *pBmp)
00661 {
00662   uint32_t height = 0, width  = 0;
00663
00664   /* Read bitmap width */
00665   width = *(uint16_t *) (pBmp + 18);
00666   width |= (*(uint16_t *) (pBmp + 20)) << 16
;
00667
00668   /* Read bitmap height */
00669   height = *(uint16_t *) (pBmp + 22);
00670   height |= (*(uint16_t *) (pBmp + 24)) << 1
6;
00671
00672   /* Remap Ypos, hx8347d works with inverted
 X in case of bitmap */
00673   /* X = 0, cursor is on Bottom corner */
00674   if(lcd_drv == &hx8347d_drv)
```

```
00675   {
00676     Ypos = BSP_LCD_GetYSize() - Ypos - heigh
t;
00677   }
00678
00679   LCD_SetDisplayWindow(Ypos, Xpos, width, he
ight);
00680
00681   if(lcd_drv->DrawBitmap != NULL)
00682   {
00683     lcd_drv->DrawBitmap(Ypos, Xpos, pBmp);
00684   }
00685   LCD_SetDisplayWindow(0, 0, BSP_LCD_GetXSize
(), BSP_LCD_GetYSize());
00686 }
00687
00688 /**
00689   * @brief  Draws a full rectangle.
00690   * @param  Xpos X position
00691   * @param  Ypos Y position
00692   * @param  Width Rectangle width
00693   * @param  Height Rectangle height
00694   * @retval None
00695   */
00696 void BSP_LCD_FillRect(uint16_t Xpos, uint16_
t Ypos, uint16_t Width, uint16_t Height)
00697 {
00698   BSP_LCD_SetTextColor(DrawProp.TextColor);
00699   do
00700   {
00701     BSP_LCD_DrawHLine(Xpos, Ypos++, Width);

00702   }
00703   while(Height--);
00704 }
00705
00706 /**
```

```
00707   * @brief  Draws a full circle.
00708   * @param  Xpos X position
00709   * @param  Ypos Y position
00710   * @param  Radius Circle radius
00711   * @retval None
00712   */
00713 void BSP_LCD_FillCircle(uint16_t Xpos, uint1
6_t Ypos, uint16_t Radius)
00714 {
00715   int32_t  decision;        /* Decision Vari
able */
00716   uint32_t  curx;    /* Current X Value */
00717   uint32_t  cury;    /* Current Y Value */
00718
00719   decision = 3 - (Radius << 1);
00720
00721   curx = 0;
00722   cury = Radius;
00723
00724   BSP_LCD_SetTextColor(DrawProp.TextColor);
00725
00726   while (curx <= cury)
00727   {
00728     if(cury > 0)
00729     {
00730       BSP_LCD_DrawVLine(Xpos + curx, Ypos -
cury, 2*cury);
00731       BSP_LCD_DrawVLine(Xpos - curx, Ypos -
cury, 2*cury);
00732     }
00733
00734     if(curx > 0)
00735     {
00736       BSP_LCD_DrawVLine(Xpos - cury, Ypos -
curx, 2*curx);
00737       BSP_LCD_DrawVLine(Xpos + cury, Ypos -
curx, 2*curx);
```

```
00738     }
00739     if (decision < 0)
00740     {
00741       decision += (curx << 2) + 6;
00742     }
00743     else
00744     {
00745       decision += ((curx - cury) << 2) + 10;
00746       cury--;
00747     }
00748     curx++;
00749   }
00750
00751   BSP_LCD_SetTextColor(DrawProp.TextColor);
00752   BSP_LCD_DrawCircle(Xpos, Ypos, Radius);
00753 }
00754
00755 /**
00756   * @brief  Draws a full ellipse.
00757   * @param  Xpos X position
00758   * @param  Ypos Y position
00759   * @param  XRadius Ellipse X radius
00760   * @param  YRadius Ellipse Y radius
00761   * @retval None
00762   */
00763 void BSP_LCD_FillEllipse(int Xpos, int Ypos,
int XRadius, int YRadius)
00764 {
00765   int x = 0, y = -XRadius, err = 2-2*YRadius
, e2;
00766   float k = 0, rad1 = 0, rad2 = 0;
00767
00768   rad1 = YRadius;
00769   rad2 = XRadius;
00770
00771   k = (float)(rad2/rad1);
00772
```

```
00773   do
00774   {
00775     BSP_LCD_DrawVLine((Xpos+y), (Ypos-(uint1
6_t)(x/k)), (2*(uint16_t)(x/k) + 1));
00776     BSP_LCD_DrawVLine((Xpos-y), (Ypos-(uint1
6_t)(x/k)), (2*(uint16_t)(x/k) + 1));
00777
00778     e2 = err;
00779     if (e2 <= x)
00780     {
00781       err += ++x*2+1;
00782       if (-y == x && e2 <= y) e2 = 0;
00783     }
00784     if (e2 > y) err += ++y*2+1;
00785   }
00786   while (y <= 0);
00787 }
00788
00789 /**
00790   * @brief  Enables the display.
00791   * @retval None
00792   */
00793 void BSP_LCD_DisplayOn(void)
00794 {
00795   lcd_drv->DisplayOn();
00796 }
00797
00798 /**
00799   * @brief  Disables the display.
00800   * @retval None
00801   */
00802 void BSP_LCD_DisplayOff(void)
00803 {
00804   lcd_drv->DisplayOff();
00805 }
00806
00807 /**
```

```
  * @}
  */

/*****************************************************************************
                                Static Function
*****************************************************************************/
/** @addtogroup STM32303C_EVAL_LCD_Private_Functions
  * @{
  */

/**
  * @brief  Draws a pixel on LCD.
  * @param  Xpos X position
  * @param  Ypos Y position
  * @param  RGBCode Pixel color in RGB mode (5-6-5)
  * @retval None
  */
static void LCD_DrawPixel(uint16_t Xpos, uint16_t Ypos, uint16_t RGBCode)
{
  if(lcd_drv->WritePixel != NULL)
  {
    lcd_drv->WritePixel(Xpos, Ypos, RGBCode);
  }
}

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

```
ata
00838   * @retval None
00839   */
00840 static void LCD_DrawChar(uint16_t Xpos, uint
16_t Ypos, const uint8_t *pChar)
00841 {
00842   uint32_t counterh = 0, counterw = 0, index
 = 0;
00843   uint16_t height = 0, width = 0;
00844   uint8_t offset = 0;
00845   uint8_t *pchar = NULL;
00846   uint32_t line = 0;
00847
00848   height = DrawProp.pFont->Height;
00849   width  = DrawProp.pFont->Width;
00850
00851   /* Fill bitmap header*/
00852   *(uint16_t *) (bitmap + 2) = (uint16_t)(he
ight*width*2+OFFSET_BITMAP);
00853   *(uint16_t *) (bitmap + 4) = (uint16_t)((h
eight*width*2+OFFSET_BITMAP)>>16);
00854   *(uint16_t *) (bitmap + 10) = OFFSET_BITMAP
;
00855   *(uint16_t *) (bitmap + 18) = (uint16_t)(w
idth);
00856   *(uint16_t *) (bitmap + 20) = (uint16_t)((
width)>>16);
00857   *(uint16_t *) (bitmap + 22) = (uint16_t)(h
eight);
00858   *(uint16_t *) (bitmap + 24) = (uint16_t)((
height)>>16);
00859
00860   offset =  8 *((width + 7)/8) -  width ;
00861
00862   for(counterh = 0; counterh < height; count
erh++)
00863   {
```

```
00864     pchar = ((uint8_t *)pChar + (width + 7)/
8 * counterh);
00865
00866     if(((width + 7)/8) == 3)
00867     {
00868       line =  (pchar[0]<< 16) | (pchar[1]<<
8) | pchar[2];
00869     }
00870
00871     if(((width + 7)/8) == 2)
00872     {
00873       line =  (pchar[0]<< 8) | pchar[1];
00874     }
00875
00876     if(((width + 7)/8) == 1)
00877     {
00878       line =  pchar[0];
00879     }
00880
00881     for (counterw = 0; counterw < width; cou
nterw++)
00882     {
00883       /* Image in the bitmap is written from
 the bottom to the top */
00884       /* Need to invert image in the bitmap
*/
00885       index = (((height-counterh-1)*width)+(
counterw))*2+OFFSET_BITMAP;
00886       if(line & (1 << (width- counterw + off
set- 1)))
00887       {
00888         bitmap[index] = (uint8_t)DrawProp.Te
xtColor;
00889         bitmap[index+1] = (uint8_t)(DrawProp.
TextColor >> 8);
00890       }
00891       else
```

```
00892       {
00893         bitmap[index] = (uint8_t)DrawProp.BackColor;
00894         bitmap[index+1] = (uint8_t)(DrawProp.BackColor >> 8);
00895       }
00896     }
00897   }
00898
00899   BSP_LCD_DrawBitmap(Ypos, Xpos, bitmap);
00900 }
00901
00902 /**
00903   * @brief  Sets display window.
00904   * @param  Xpos LCD X position
00905   * @param  Ypos LCD Y position
00906   * @param  Width LCD window width
00907   * @param  Height LCD window height
00908   * @retval None
00909   */
00910 static void LCD_SetDisplayWindow(uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height)
00911 {
00912   if(lcd_drv->SetDisplayWindow != NULL)
00913   {
00914     lcd_drv->SetDisplayWindow(Xpos, Ypos, Width, Height);
00915   }
00916 }
00917 /**
00918   * @}
00919   */
00920
00921 /**
00922   * @}
00923   */
```

```
00924
00925 /**
00926   * @}
00927   */
00928
00929 /**
00930   * @}
00931   */
00932
00933 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM32303C_EVAL BSP User Manual

Exported Constants

STM32303C-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 157 of file stm32303c_eval_lcd.h.

#define LCD_COLOR_BLUE 0x001F

LCD color.

Definition at line 135 of file stm32303c_eval_lcd.h.

#define LCD_COLOR_BROWN 0xA145

Definition at line 158 of file stm32303c_eval_lcd.h.

#define LCD_COLOR_CYAN 0x07FF

Definition at line 138 of file stm32303c_eval_lcd.h.

#define LCD_COLOR_DARKBLUE 0x0010

Definition at line 147 of file stm32303c_eval_lcd.h.

#define LCD_COLOR_DARKCYAN 0x0410

Definition at line 150 of file stm32303c_eval_lcd.h.

#define LCD_COLOR_DARKGRAY 0x4208

Definition at line **156** of file **stm32303c_eval_lcd.h**.

#define LCD_COLOR_DARKGREEN 0x0400

Definition at line **148** of file **stm32303c_eval_lcd.h**.

#define LCD_COLOR_DARKMAGENTA 0x8010

Definition at line **151** of file **stm32303c_eval_lcd.h**.

#define LCD_COLOR_DARKRED 0x8000

Definition at line **149** of file **stm32303c_eval_lcd.h**.

#define LCD_COLOR_DARKYELLOW 0x8400

Definition at line **152** of file **stm32303c_eval_lcd.h**.

#define LCD_COLOR_GRAY 0x8410

Definition at line **155** of file **stm32303c_eval_lcd.h**.

#define LCD_COLOR_GREEN 0x07E0

Definition at line **136** of file **stm32303c_eval_lcd.h**.

#define LCD_COLOR_LIGHTBLUE 0x841F

Definition at line **141** of file **stm32303c_eval_lcd.h**.

#define LCD_COLOR_LIGHTCYAN 0x87FF

Definition at line 144 of file stm32303c_eval_lcd.h.

#define LCD_COLOR_LIGHTGRAY 0xD69A

Definition at line 154 of file stm32303c_eval_lcd.h.

#define LCD_COLOR_LIGHTGREEN 0x87F0

Definition at line 142 of file stm32303c_eval_lcd.h.

#define LCD_COLOR_LIGHTMAGENTA 0xFC1F

Definition at line 145 of file stm32303c_eval_lcd.h.

#define LCD_COLOR_LIGHTRED 0xFC10

Definition at line 143 of file stm32303c_eval_lcd.h.

#define LCD_COLOR_LIGHTYELLOW 0xFFF0

Definition at line 146 of file stm32303c_eval_lcd.h.

#define LCD_COLOR_MAGENTA 0xF81F

Definition at line 139 of file stm32303c_eval_lcd.h.

#define LCD_COLOR_ORANGE 0xFD20

Definition at line 159 of file stm32303c_eval_lcd.h.

#define LCD_COLOR_RED 0xF800

Definition at line 137 of file stm32303c_eval_lcd.h.

#define LCD_COLOR_WHITE 0xFFFF

Definition at line 153 of file stm32303c_eval_lcd.h.

#define LCD_COLOR_YELLOW 0xFFE0

Definition at line 140 of file stm32303c_eval_lcd.h.

#define LCD_DEFAULT_FONT Font24

LCD default font.

Definition at line 164 of file stm32303c_eval_lcd.h.

Referenced by BSP_LCD_DrawCircle(), and BSP_LCD_Init().

#define LCD_ERROR 0x01

Definition at line 111 of file stm32303c_eval_lcd.h.

Referenced by BSP_LCD_Init().

#define LCD_OK 0x00

LCD status structure definition.

Definition at line **110** of file **stm32303c_eval_lcd.h**.

Referenced by **BSP_LCD_Init()**.

#define LCD_TIMEOUT 0x02

Definition at line **112** of file **stm32303c_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 **124** of file **stm32303c_eval_lcd.h**.

STM32303C_EVAL BSP User Manual

STM32303C-EVAL SD

STM32303C-EVAL

Modules

- Types Definitions
- Private Constants
- Private Variables
- Exported Types
- Exported Constants
- Exported Macro
- Exported Functions
- Private Functions

STM32303C_EVAL BSP User Manual

Exported Types

STM32303C-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 85 of file stm32303c_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 84 of file stm32303c_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 **87** of file **stm32303c_eval_sd.h**.

STM32303C_EVAL BSP User Manual

stm32303c_eval_sd.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32303c_eval_sd.h
00004   * @author  MCD Application Team
00005   * @brief   This file contains the common defines and functions prototypes for
00006   *          the stm32303c_eval_sd.c driver.
00007   ******************************************************************************
00008   * @attention
00009   *
00010   * <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>
00011   *
00012   * Redistribution and use in source and binary forms, with or without modification,
00013   * are permitted provided that the following conditions are met:
00014   *   1. Redistributions of source code must retain the above copyright notice,
00015   *      this list of conditions and the fol
```

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

```
00035   */
00036
00037 /* Define to prevent recursive inclusion ---
-------------------------------------*/
00038 #ifndef __STM32303C_EVAL_SD_H
00039 #define __STM32303C_EVAL_SD_H
00040
00041 #ifdef __cplusplus
00042  extern "C" {
00043 #endif
00044
00045 /* Includes ------------------------------
-------------------------------------*/
00046 #include "stm32303c_eval.h"
00047
00048 /** @addtogroup BSP
00049   * @{
00050   */
00051
00052 /** @addtogroup STM32303C_EVAL
00053   * @{
00054   */
00055
00056 /** @defgroup STM32303C_EVAL_SD STM32303C-EV
AL SD
00057   * @{
00058   */
00059
00060 /* Private define ------------------------
-------------------------------------*/
00061
00062 /** @defgroup STM32303C_EVAL_SD_Private_Defi
nes Private Constants
00063  * @{
00064  */
00065 /**
00066  * @}
```

```
00067  */
00068 /* Private variables ---------------------
-----------------------------------*/
00069
00070 /** @defgroup STM32303C_EVAL_SD_Private_Vari
ables Private Variables
00071  * @{
00072  */
00073 /**
00074  * @}
00075  */
00076
00077 /** @defgroup STM32303C_EVAL_SD_Exported_Typ
es Exported Types
00078   * @{
00079   */
00080
00081 /**
00082   * @brief  SD status structure definition
00083   */
00084 #define MSD_OK          0x00
00085 #define MSD_ERROR       0x01
00086
00087 typedef enum
00088 {
00089 /**
00090   * @brief  SD reponses and error flags
00091   */
00092   SD_RESPONSE_NO_ERROR      = (0x00),
00093   SD_IN_IDLE_STATE          = (0x01),
00094   SD_ERASE_RESET            = (0x02),
00095   SD_ILLEGAL_COMMAND        = (0x04),
00096   SD_COM_CRC_ERROR          = (0x08),
00097   SD_ERASE_SEQUENCE_ERROR   = (0x10),
00098   SD_ADDRESS_ERROR          = (0x20),
00099   SD_PARAMETER_ERROR        = (0x40),
00100   SD_RESPONSE_FAILURE       = (0xFF),
```

```
00101
00102 /**
00103   * @brief  Data response error
00104   */
00105   SD_DATA_OK                = (0x05),
00106   SD_DATA_CRC_ERROR         = (0x0B),
00107   SD_DATA_WRITE_ERROR       = (0x0D),
00108   SD_DATA_OTHER_ERROR       = (0xFF)
00109 }SD_Info;
00110
00111 /**
00112   * @brief  Card Specific Data: CSD Register
00113   */
00114 typedef struct
00115 {
00116   __IO uint8_t  CSDStruct;            /* CSD
 structure */
00117   __IO uint8_t  SysSpecVersion;       /* Sys
tem specification version */
00118   __IO uint8_t  Reserved1;            /* Res
erved */
00119   __IO uint8_t  TAAC;                 /* Dat
a read access-time 1 */
00120   __IO uint8_t  NSAC;                 /* Dat
a read access-time 2 in CLK cycles */
00121   __IO uint8_t  MaxBusClkFrec;        /* Max
. bus clock frequency */
00122   __IO uint16_t CardComdClasses;      /* Car
d command classes */
00123   __IO uint8_t  RdBlockLen;           /* Max
. read data block length */
00124   __IO uint8_t  PartBlockRead;        /* Par
tial blocks for read allowed */
00125   __IO uint8_t  WrBlockMisalign;      /* Wri
te block misalignment */
00126   __IO uint8_t  RdBlockMisalign;      /* Rea
d block misalignment */
```

```
00127   __IO uint8_t  DSRImpl;                /* DSR
 implemented */
00128   __IO uint8_t  Reserved2;              /* Res
erved */
00129   __IO uint32_t DeviceSize;             /* Dev
ice Size */
00130   __IO uint8_t  MaxRdCurrentVDDMin;     /* Max
. read current @ VDD min */
00131   __IO uint8_t  MaxRdCurrentVDDMax;     /* Max
. read current @ VDD max */
00132   __IO uint8_t  MaxWrCurrentVDDMin;     /* Max
. write current @ VDD min */
00133   __IO uint8_t  MaxWrCurrentVDDMax;     /* Max
. write current @ VDD max */
00134   __IO uint8_t  DeviceSizeMul;          /* Dev
ice size multiplier */
00135   __IO uint8_t  EraseGrSize;            /* Era
se group size */
00136   __IO uint8_t  EraseGrMul;             /* Era
se group size multiplier */
00137   __IO uint8_t  WrProtectGrSize;        /* Wri
te protect group size */
00138   __IO uint8_t  WrProtectGrEnable;      /* Wri
te protect group enable */
00139   __IO uint8_t  ManDeflECC;             /* Man
ufacturer default ECC */
00140   __IO uint8_t  WrSpeedFact;            /* Wri
te speed factor */
00141   __IO uint8_t  MaxWrBlockLen;          /* Max
. write data block length */
00142   __IO uint8_t  WriteBlockPaPartial;    /* Par
tial blocks for write allowed */
00143   __IO uint8_t  Reserved3;              /* Res
erded */
00144   __IO uint8_t  ContentProtectAppli;    /* Con
tent protection application */
00145   __IO uint8_t  FileFormatGrouop;       /* Fil
```

```
e format group */
00146   __IO uint8_t  CopyFlag;             /* Cop
y flag (OTP) */
00147   __IO uint8_t  PermWrProtect;        /* Per
manent write protection */
00148   __IO uint8_t  TempWrProtect;        /* Tem
porary write protection */
00149   __IO uint8_t  FileFormat;           /* Fil
e Format */
00150   __IO uint8_t  ECC;                  /* ECC
 code */
00151   __IO uint8_t  CSD_CRC;              /* CSD
 CRC */
00152   __IO uint8_t  Reserved4;            /* alw
ays 1*/
00153 } SD_CSD;
00154
00155 /**
00156   * @brief  Card Identification Data: CID Re
gister
00157   */
00158 typedef struct
00159 {
00160   __IO uint8_t  ManufacturerID;       /* Man
ufacturerID */
00161   __IO uint16_t OEM_AppliID;          /* OEM
/Application ID */
00162   __IO uint32_t ProdName1;            /* Pro
duct Name part1 */
00163   __IO uint8_t  ProdName2;            /* Pro
duct Name part2*/
00164   __IO uint8_t  ProdRev;              /* Pro
duct Revision */
00165   __IO uint32_t ProdSN;               /* Pro
duct Serial Number */
00166   __IO uint8_t  Reserved1;            /* Res
erved1 */
```

```
00167   __IO uint16_t ManufactDate;          /* Man
ufacturing Date */
00168   __IO uint8_t  CID_CRC;               /* CID
 CRC */
00169   __IO uint8_t  Reserved2;             /* alw
ays 1 */
00170 } SD_CID;
00171
00172 /**
00173   * @brief SD Card information
00174   */
00175 typedef struct
00176 {
00177   SD_CSD Csd;
00178   SD_CID Cid;
00179   uint32_t CardCapacity;  /* Card Capacity */

00180   uint32_t CardBlockSize; /* Card Block Size
 */
00181 } SD_CardInfo;
00182
00183 /**
00184   * @}
00185   */
00186
00187 /** @defgroup STM32303C_EVAL_SD_Exported_Con
stants Exported Constants
00188   * @{
00189   */
00190
00191 /**
00192   * @brief  Start Data tokens:
00193   *         Tokens (necessary because at nop
/idle (and CS active) only 0xff is
00194   *         on the data/command line)
00195   */
00196 #define SD_START_DATA_SINGLE_BLOCK_READ    0
```

```
xFE  /* Data token start byte, Start Single Block
Read */
00197 #define SD_START_DATA_MULTIPLE_BLOCK_READ  0
xFE  /* Data token start byte, Start Multiple Bloc
k Read */
00198 #define SD_START_DATA_SINGLE_BLOCK_WRITE   0
xFE  /* Data token start byte, Start Single Block
Write */
00199 #define SD_START_DATA_MULTIPLE_BLOCK_WRITE 0
xFD  /* Data token start byte, Start Multiple Bloc
k Write */
00200 #define SD_STOP_DATA_MULTIPLE_BLOCK_WRITE  0
xFD  /* Data toke stop byte, Stop Multiple Block W
rite */
00201
00202 /**
00203   * @brief  SD detection on its memory slot
00204   */
00205 #define SD_PRESENT               ((uint8_t)0
x01)
00206 #define SD_NOT_PRESENT           ((uint8_t)0
x00)
00207
00208 /**
00209   * @brief  Commands: CMDxx = CMD-number | 0
x40
00210   */
00211 #define SD_CMD_GO_IDLE_STATE          0   /*
 CMD0 = 0x40 */
00212 #define SD_CMD_SEND_OP_COND           1   /*
 CMD1 = 0x41 */
00213 #define SD_CMD_SEND_CSD               9   /*
 CMD9 = 0x49 */
00214 #define SD_CMD_SEND_CID               10  /*
 CMD10 = 0x4A */
00215 #define SD_CMD_STOP_TRANSMISSION      12  /*
 CMD12 = 0x4C */
```

```
00216 #define SD_CMD_SEND_STATUS              13  /*
 CMD13 = 0x4D */
00217 #define SD_CMD_SET_BLOCKLEN             16  /*
 CMD16 = 0x50 */
00218 #define SD_CMD_READ_SINGLE_BLOCK        17  /*
 CMD17 = 0x51 */
00219 #define SD_CMD_READ_MULT_BLOCK          18  /*
 CMD18 = 0x52 */
00220 #define SD_CMD_SET_BLOCK_COUNT          23  /*
 CMD23 = 0x57 */
00221 #define SD_CMD_WRITE_SINGLE_BLOCK       24  /*
 CMD24 = 0x58 */
00222 #define SD_CMD_WRITE_MULT_BLOCK         25  /*
 CMD25 = 0x59 */
00223 #define SD_CMD_PROG_CSD                 27  /*
 CMD27 = 0x5B */
00224 #define SD_CMD_SET_WRITE_PROT           28  /*
 CMD28 = 0x5C */
00225 #define SD_CMD_CLR_WRITE_PROT           29  /*
 CMD29 = 0x5D */
00226 #define SD_CMD_SEND_WRITE_PROT          30  /*
 CMD30 = 0x5E */
00227 #define SD_CMD_SD_ERASE_GRP_START       32  /*
 CMD32 = 0x60 */
00228 #define SD_CMD_SD_ERASE_GRP_END         33  /*
 CMD33 = 0x61 */
00229 #define SD_CMD_UNTAG_SECTOR             34  /*
 CMD34 = 0x62 */
00230 #define SD_CMD_ERASE_GRP_START          35  /*
 CMD35 = 0x63 */
00231 #define SD_CMD_ERASE_GRP_END            36  /*
 CMD36 = 0x64 */
00232 #define SD_CMD_UNTAG_ERASE_GROUP        37  /*
 CMD37 = 0x65 */
00233 #define SD_CMD_ERASE                    38  /*
 CMD38 = 0x66 */
00234
```

```
00235 /**
00236   * @}
00237   */
00238
00239 /** @defgroup STM32303C_EVAL_SD_Exported_Mac
ro Exported Macro
00240   * @{
00241   */
00242
00243 /**
00244   * @}
00245   */
00246
00247 /** @defgroup STM32303C_EVAL_SD_Exported_Fun
ctions Exported Functions
00248   * @{
00249   */
00250 uint8_t BSP_SD_Init(void);
00251 uint8_t  BSP_SD_IsDetected(void);
00252 uint8_t BSP_SD_ReadBlocks(uint32_t* p32Data,
 uint64_t ReadAddr, uint16_t BlockSize, uint32_t N
umberOfBlocks);
00253 uint8_t BSP_SD_WriteBlocks(uint32_t* p32Data
, uint64_t WriteAddr, uint16_t BlockSize, uint32_t
 NumberOfBlocks);
00254 uint8_t BSP_SD_Erase(uint32_t StartAddr, uin
t32_t EndAddr);
00255 uint8_t BSP_SD_GetStatus(void);
00256 uint8_t BSP_SD_GetCardInfo(SD_CardInfo *pCar
dInfo);
00257
00258 /* Link functions for SD Card peripheral*/
00259 void                  SD_IO_Init(void);
00260 void                  SD_IO_WriteByte(uint
8_t Data);
00261 uint8_t               SD_IO_ReadByte(void)
;
```

```
00262 HAL_StatusTypeDef       SD_IO_WriteCmd(uint8
_t Cmd, uint32_t Arg, uint8_t Crc, uint8_t Respons
e);
00263 HAL_StatusTypeDef       SD_IO_WaitResponse(u
int8_t Response);
00264 void                    SD_IO_WriteDummy(void
);
00265
00266 /**
00267   * @}
00268   */
00269
00270 /** @defgroup STM32303C_EVAL_SD_Private_Func
tions Private Functions
00271   * @{
00272   */
00273
00274 /**
00275   * @}
00276   */
00277
00278 /**
00279   * @}
00280   */
00281
00282 /**
00283   * @}
00284   */
00285
00286 /**
00287   * @}
00288   */
00289
00290 #ifdef __cplusplus
00291 }
00292 #endif
00293
```

```
00294 #endif /* __H */
00295
00296 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM32303C_EVAL BSP User Manual

stm32303c_eval_sd.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32303c_eval_sd.c
00004   * @author  MCD Application Team
00005   * @brief   This file provides a set of functions needed to manage the SPI SD
00006   *          Card memory mounted on STM32303C-EVAL board.
00007   *          It implements a high level communication layer for read and write
00008   *          from/to this memory. The needed STM32F30x hardware resources (SPI and
00009   *          GPIO) are defined in stm32303c_eval.h file, and the initialization is
00010   *          performed in SD_LowLevel_Init() function declared in stm32303c_eval.c
00011   *          file.
00012   *          You can easily tailor this driver to any other development board,
00013   *          by just adapting the defines for hardware resources and
```

```
*          SD_LowLevel_Init() function.
*
*          +-----------------------------------------------------+
*          |                    Pin assignment                   |
*          +-------------------------+-------------+-------------+
*          |  STM32F30x SPI Pins     |  SD         |    Pin      |
*          +-------------------------+-------------+-------------+
*          | SD_SPI_CS_PIN           |   ChipSelect  |    1        |
*          | SD_SPI_MOSI_PIN / MOSI  |   DataIn      |    2        |
*          |                         |   GND         |    3 (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
*
* <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>
*
* Redistribution and use in source and binary forms, with or without modification,
```

```
00035   * are permitted provided that the followin
g conditions are met:
00036   *   1. Redistributions of source code must
 retain the above copyright notice,
00037   *      this list of conditions and the fol
lowing disclaimer.
00038   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00039   *      this list of conditions and the fol
lowing disclaimer in the documentation
00040   *      and/or other materials provided wit
h the distribution.
00041   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00042   *      may be used to endorse or promote p
roducts derived from this software
00043   *      without specific prior written perm
ission.
00044   *
00045   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00046   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00047   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00048   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00049   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00050   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 PROCUREMENT OF SUBSTITUTE GOODS OR
00051   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00052   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00053   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
```

```
00054   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00055   *
00056   ****************************************
**********************************
00057   */
00058
00059 /* File Info : ------------------------------
-------------------------------------
00060                                       User NOTES

00061 1. How To use this driver:
00062 --------------------------
00063    - This driver is used to drive the micro
SD external card mounted on STM32303C-EVAL
00064      evaluation board.
00065    - This driver does not need a specific co
mponent driver for the micro SD device
00066      to be included with.
00067
00068 2. Driver description:
00069 ---------------------
00070   + Initialization steps:
00071      o Initialize the micro SD card using th
e SD_Init() function.
00072      o To check the SD card presence you can
 use the function SD_IsDetected() which
00073        returns the detection status
00074      o The function SD_GetCardInfo() is used
 to get the micro SD card information
00075        which is stored in the structure "HAL
_SD_CardInfoTypedef".
00076
00077   + Micro SD card operations
00078      o The micro SD card can be accessed wit
h read/write block(s) operations once
00079        it is reay for access. The access can
```

```
d be performed in polling
00080        mode by calling the functions SD_Read
Blocks()/SD_WriteBlocks()
00081      o The SD erase block(s) is performed us
ing the function SD_Erase() with specifying
00082        the number of blocks to erase.
00083      o The SD runtime status is returned whe
n calling the function SD_GetStatus().
00084
00085 ----------------------------------------
----------------------------------*/
00086
00087 /* Includes ------------------------------
----------------------------------*/
00088 #include "stm32303c_eval_sd.h"
00089
00090 /** @addtogroup BSP
00091   * @{
00092   */
00093
00094 /** @addtogroup STM32303C_EVAL
00095   * @{
00096   */
00097
00098 /** @addtogroup STM32303C_EVAL_SD
00099   * @{
00100   */
00101
00102 /* Private typedef -----------------------
----------------------------------*/
00103
00104 /** @defgroup STM32303C_EVAL_SD_Private_Type
s_Definitions Types Definitions
00105   * @{
00106   */
00107
00108 /**
```

```
00109   * @}
00110   */
00111 /* Private define ------------------------
-----------------------------------*/
00112
00113 /** @addtogroup STM32303C_EVAL_SD_Private_De
fines
00114   * @{
00115   */
00116 #define SD_DUMMY_BYTE   0xFF
00117 #define SD_NO_RESPONSE_EXPECTED 0x80
00118 /**
00119   * @}
00120   */
00121
00122 /* Private macro ------------------------
-----------------------------------*/
00123
00124 /* Private variables ---------------------
-----------------------------------*/
00125
00126 /** @addtogroup STM32303C_EVAL_SD_Private_Va
riables
00127   * @{
00128   */
00129 __IO uint8_t SdStatus = SD_PRESENT;
00130
00131 /**
00132   * @}
00133   */
00134
00135 /* Private function prototypes ------------
-----------------------------------*/
00136 /** @addtogroup STM32303C_EVAL_SD_Private_Fu
nctions
00137  * @{
00138  */
```

```
00139 static uint8_t SD_GetCIDRegister(SD_CID* Cid
);
00140 static uint8_t SD_GetCSDRegister(SD_CSD* Csd
);
00141 static SD_Info SD_GetDataResponse(void);
00142 static uint8_t SD_GoIdleState(void);
00143 static uint8_t SD_SendCmd(uint8_t Cmd, uint3
2_t Arg, uint8_t Crc, uint8_t Response);
00144 /* Private functions ---------------------
-----------------------------------*/
00145
00146 /**
00147   * @brief  Initializes the SD/SD communicat
ion.
00148   * @retval The SD Response:
00149   *         - MSD_ERROR : Sequence failed
00150   *         - MSD_OK    : Sequence succeed
00151   */
00152 uint8_t BSP_SD_Init(void)
00153 {
00154   /* Configure IO functionalities for SD pin
 */
00155   SD_IO_Init();
00156
00157   /* Check SD card detect pin */
00158   if(BSP_SD_IsDetected()==SD_NOT_PRESENT)
00159   {
00160     SdStatus = SD_NOT_PRESENT;
00161     return MSD_ERROR;
00162   }
00163   else
00164   {
00165     SdStatus = SD_PRESENT;
00166   }
00167
00168   /* SD initialized and set to SPI mode prop
erly */
```

```
00169   return (SD_GoIdleState());
00170 }
00171
00172 /**
00173  * @brief  Detects if SD card is correctly p
lugged in the memory slot or not.
00174  * @retval Returns if SD is detected or not
00175  */
00176 uint8_t BSP_SD_IsDetected(void)
00177 {
00178   __IO uint8_t status = SD_PRESENT;
00179
00180   /* Check SD card detect pin */
00181   if(HAL_GPIO_ReadPin(SD_DETECT_GPIO_PORT, S
D_DETECT_PIN) != GPIO_PIN_RESET)
00182   {
00183     status = SD_NOT_PRESENT;
00184   }
00185
00186   return status;
00187 }
00188
00189 /**
00190   * @brief  Returns information about specif
ic card.
00191   * @param  pCardInfo pointer to a SD_CardIn
fo structure that contains all SD
00192   *         card information.
00193   * @retval The SD Response:
00194   *         - MSD_ERROR : Sequence failed
00195   *         - MSD_OK    : Sequence succeed
00196   */
00197 uint8_t BSP_SD_GetCardInfo(SD_CardInfo *pCar
dInfo)
00198 {
00199   uint8_t status = MSD_ERROR;
00200
```

```
00201   SD_GetCSDRegister(&(pCardInfo->Csd));
00202   status = SD_GetCIDRegister(&(pCardInfo->Cid
));
00203   pCardInfo->CardCapacity = (pCardInfo->Csd.
DeviceSize + 1) ;
00204   pCardInfo->CardCapacity *= (1 << (pCardInf
o->Csd.DeviceSizeMul + 2));
00205   pCardInfo->CardBlockSize = 1 << (pCardInfo
->Csd.RdBlockLen);
00206   pCardInfo->CardCapacity *= pCardInfo->Card
BlockSize;
00207
00208   /* Returns the reponse */
00209   return status;
00210 }
00211
00212 /**
00213   * @brief  Reads block(s) from a specified
address in an SD card, in polling mode.
00214   * @param  p32Data Pointer to the buffer th
at will contain the data to transmit
00215   * @param  ReadAddr Address from where data
 is to be read
00216   * @param  BlockSize SD card data block siz
e, that should be 512
00217   * @param  NumberOfBlocks Number of SD bloc
ks to read
00218   * @retval SD status
00219   */
00220 uint8_t BSP_SD_ReadBlocks(uint32_t* p32Data,
 uint64_t ReadAddr, uint16_t BlockSize, uint32_t N
umberOfBlocks)
00221 {
00222   uint32_t counter = 0, offset = 0;
00223   uint8_t rvalue = MSD_ERROR;
00224   uint8_t *pData = (uint8_t *)p32Data;
00225
```

```
00226   /* Send CMD16 (SD_CMD_SET_BLOCKLEN) to set
 the size of the block and
00227      Check if the SD acknowledged the set bl
ock length command: R1 response (0x00: no errors)
*/
00228   if (SD_IO_WriteCmd(SD_CMD_SET_BLOCKLEN, Bl
ockSize, 0xFF, SD_RESPONSE_NO_ERROR) != HAL_OK)
00229   {
00230     return MSD_ERROR;
00231   }
00232
00233   /* Data transfer */
00234   while (NumberOfBlocks--)
00235   {
00236     /* Send dummy byte: 8 Clock pulses of de
lay */
00237     SD_IO_WriteDummy();
00238
00239     /* Send CMD17 (SD_CMD_READ_SINGLE_BLOCK)
 to read one block */
00240     /* Check if the SD acknowledged the read
 block command: R1 response (0x00: no errors) */
00241     if (SD_IO_WriteCmd(SD_CMD_READ_SINGLE_BL
OCK, ReadAddr + offset, 0xFF, SD_RESPONSE_NO_ERROR
) != HAL_OK)
00242     {
00243       return MSD_ERROR;
00244     }
00245
00246     /* Now look for the data token to signif
y the start of the data */
00247     if (SD_IO_WaitResponse(SD_START_DATA_SIN
GLE_BLOCK_READ) == HAL_OK)
00248     {
00249       /* Read the SD block data : read NumBy
teToRead data */
00250       for (counter = 0; counter < BlockSize;
```

```
 counter++)
00251       {
00252         /* Read the pointed data */
00253         *pData = SD_IO_ReadByte();
00254         /* Point to the next location where
the byte read will be saved */
00255         pData++;
00256       }
00257       /* Set next read address*/
00258       offset += BlockSize;
00259       /* get CRC bytes (not really needed by
 us, but required by SD) */
00260       SD_IO_ReadByte();
00261       SD_IO_ReadByte();
00262       /* Set response value to success */
00263       rvalue = MSD_OK;
00264     }
00265     else
00266     {
00267       /* Set response value to failure */
00268       rvalue = MSD_ERROR;
00269     }
00270   }
00271
00272   /* Send dummy byte: 8 Clock pulses of dela
y */
00273   SD_IO_WriteDummy();
00274   /* Returns the reponse */
00275   return rvalue;
00276 }
00277
00278 /**
00279   * @brief  Writes block(s) to a specified a
ddress in an SD card, in polling mode.
00280   * @param  p32Data Pointer to the buffer th
at will contain the data to transmit
00281   * @param  WriteAddr Address from where dat
```

```
a is to be written
00282   * @param  BlockSize SD card data block siz
e, that should be 512
00283   * @param  NumberOfBlocks Number of SD bloc
ks to write
00284   * @retval SD status
00285   */
00286 uint8_t BSP_SD_WriteBlocks(uint32_t* p32Data
, uint64_t WriteAddr, uint16_t BlockSize, uint32_t
 NumberOfBlocks)
00287 {
00288   uint32_t counter = 0, offset = 0;
00289   uint8_t rvalue = MSD_ERROR;
00290   uint8_t *pData = (uint8_t *)p32Data;
00291
00292   /* Data transfer */
00293   while (NumberOfBlocks--)
00294   {
00295     /* Send CMD24 (SD_CMD_WRITE_SINGLE_BLOCK
) to write blocks  and
00296        Check if the SD acknowledged the writ
e block command: R1 response (0x00: no errors) */
00297     if (SD_IO_WriteCmd(SD_CMD_WRITE_SINGLE_B
LOCK, WriteAddr + offset, 0xFF, SD_RESPONSE_NO_ERR
OR) != HAL_OK)
00298     {
00299       return MSD_ERROR;
00300     }
00301
00302     /* Send dummy byte */
00303     SD_IO_WriteByte(SD_DUMMY_BYTE);
00304
00305     /* Send the data token to signify the st
art of the data */
00306     SD_IO_WriteByte(SD_START_DATA_SINGLE_BLO
CK_WRITE);
00307
```

```
00308     /* Write the block data to SD : write co
unt data by block */
00309     for (counter = 0; counter < BlockSize; c
ounter++)
00310     {
00311       /* Send the pointed byte */
00312       SD_IO_WriteByte(*pData);
00313
00314       /* Point to the next location where th
e byte read will be saved */
00315       pData++;
00316     }
00317
00318     /* Set next write address */
00319     offset += BlockSize;
00320
00321     /* Put CRC bytes (not really needed by u
s, but required by SD) */
00322     SD_IO_ReadByte();
00323     SD_IO_ReadByte();
00324
00325     /* Read data response */
00326     if (SD_GetDataResponse() == SD_DATA_OK)
00327     {
00328       /* Set response value to success */
00329       rvalue = MSD_OK;
00330     }
00331     else
00332     {
00333       /* Set response value to failure */
00334       rvalue = MSD_ERROR;
00335     }
00336   }
00337
00338   /* Send dummy byte: 8 Clock pulses of dela
y */
00339   SD_IO_WriteDummy();
```

```
00340
00341   /* Returns the reponse */
00342   return rvalue;
00343 }
00344
00345 /**
00346   * @brief  Read the CSD card register.
00347   *         Reading the contents of the CSD
register in SPI mode is a simple
00348   *         read-block transaction.
00349   * @param  Csd pointer on an SCD register s
tructure
00350   * @retval SD status
00351   */
00352 uint8_t SD_GetCSDRegister(SD_CSD* Csd)
00353 {
00354   uint32_t counter = 0;
00355   uint8_t rvalue = MSD_ERROR;
00356   uint8_t CSD_Tab[16];
00357
00358   /* Send CMD9 (CSD register) or CMD10(CSD r
egister) and Wait for response in the R1 format (0
x00 is no errors) */
00359   if (SD_IO_WriteCmd(SD_CMD_SEND_CSD, 0, 0xF
F, SD_RESPONSE_NO_ERROR) == HAL_OK)
00360   {
00361     if (SD_IO_WaitResponse(SD_START_DATA_SIN
GLE_BLOCK_READ) == HAL_OK)
00362     {
00363       for (counter = 0; counter < 16; counte
r++)
00364       {
00365         /* Store CSD register value on CSD_T
ab */
00366         CSD_Tab[counter] = SD_IO_ReadByte();
00367       }
00368
```

```
00369       /* Get CRC bytes (not really needed by
 us, but required by SD) */
00370       SD_IO_WriteByte(SD_DUMMY_BYTE);
00371       SD_IO_WriteByte(SD_DUMMY_BYTE);
00372
00373       /* Set response value to success */
00374       rvalue = MSD_OK;
00375     }
00376   }
00377   /* Send dummy byte: 8 Clock pulses of dela
y */
00378   SD_IO_WriteDummy();
00379
00380   if(rvalue == SD_RESPONSE_NO_ERROR)
00381   {
00382     /* Byte 0 */
00383     Csd->CSDStruct = (CSD_Tab[0] & 0xC0) >>
6;
00384     Csd->SysSpecVersion = (CSD_Tab[0] & 0x3C
) >> 2;
00385     Csd->Reserved1 = CSD_Tab[0] & 0x03;
00386
00387     /* Byte 1 */
00388     Csd->TAAC = CSD_Tab[1];
00389
00390     /* Byte 2 */
00391     Csd->NSAC = CSD_Tab[2];
00392
00393     /* Byte 3 */
00394     Csd->MaxBusClkFrec = CSD_Tab[3];
00395
00396     /* Byte 4 */
00397     Csd->CardComdClasses = CSD_Tab[4] << 4;
00398
00399     /* Byte 5 */
00400     Csd->CardComdClasses |= (CSD_Tab[5] & 0x
F0) >> 4;
```

```
00401     Csd->RdBlockLen = CSD_Tab[5] & 0x0F;
00402
00403     /* Byte 6 */
00404     Csd->PartBlockRead = (CSD_Tab[6] & 0x80)
 >> 7;
00405     Csd->WrBlockMisalign = (CSD_Tab[6] & 0x4
0) >> 6;
00406     Csd->RdBlockMisalign = (CSD_Tab[6] & 0x2
0) >> 5;
00407     Csd->DSRImpl = (CSD_Tab[6] & 0x10) >> 4;
00408     Csd->Reserved2 = 0; /*!< Reserved */
00409
00410     Csd->DeviceSize = (CSD_Tab[6] & 0x03) <<
 10;
00411
00412     /* Byte 7 */
00413     Csd->DeviceSize |= (CSD_Tab[7]) << 2;
00414
00415     /* Byte 8 */
00416     Csd->DeviceSize |= (CSD_Tab[8] & 0xC0) >
> 6;
00417
00418     Csd->MaxRdCurrentVDDMin = (CSD_Tab[8] &
0x38) >> 3;
00419     Csd->MaxRdCurrentVDDMax = (CSD_Tab[8] &
0x07);
00420
00421     /* Byte 9 */
00422     Csd->MaxWrCurrentVDDMin = (CSD_Tab[9] &
0xE0) >> 5;
00423     Csd->MaxWrCurrentVDDMax = (CSD_Tab[9] &
0x1C) >> 2;
00424     Csd->DeviceSizeMul = (CSD_Tab[9] & 0x03)
 << 1;
00425     /* Byte 10 */
00426     Csd->DeviceSizeMul |= (CSD_Tab[10] & 0x8
0) >> 7;
```

```
00427
00428     Csd->EraseGrSize = (CSD_Tab[10] & 0x40)
>> 6;
00429     Csd->EraseGrMul = (CSD_Tab[10] & 0x3F) <
< 1;
00430
00431     /* Byte 11 */
00432     Csd->EraseGrMul |= (CSD_Tab[11] & 0x80)
>> 7;
00433     Csd->WrProtectGrSize = (CSD_Tab[11] & 0x
7F);
00434
00435     /* Byte 12 */
00436     Csd->WrProtectGrEnable = (CSD_Tab[12] &
0x80) >> 7;
00437     Csd->ManDeflECC = (CSD_Tab[12] & 0x60) >
> 5;
00438     Csd->WrSpeedFact = (CSD_Tab[12] & 0x1C)
>> 2;
00439     Csd->MaxWrBlockLen = (CSD_Tab[12] & 0x03
) << 2;
00440
00441     /* Byte 13 */
00442     Csd->MaxWrBlockLen |= (CSD_Tab[13] & 0xC
0) >> 6;
00443     Csd->WriteBlockPaPartial = (CSD_Tab[13]
& 0x20) >> 5;
00444     Csd->Reserved3 = 0;
00445     Csd->ContentProtectAppli = (CSD_Tab[13]
& 0x01);
00446
00447     /* Byte 14 */
00448     Csd->FileFormatGrouop = (CSD_Tab[14] & 0
x80) >> 7;
00449     Csd->CopyFlag = (CSD_Tab[14] & 0x40) >>
6;
00450     Csd->PermWrProtect = (CSD_Tab[14] & 0x20
```

```
) >> 5;
00451     Csd->TempWrProtect = (CSD_Tab[14] & 0x10
) >> 4;
00452     Csd->FileFormat = (CSD_Tab[14] & 0x0C) >
> 2;
00453     Csd->ECC = (CSD_Tab[14] & 0x03);
00454
00455     /* Byte 15 */
00456     Csd->CSD_CRC = (CSD_Tab[15] & 0xFE) >> 1
;
00457     Csd->Reserved4 = 1;
00458   }
00459   /* Return the reponse */
00460   return rvalue;
00461 }
00462
00463 /**
00464   * @brief  Read the CID card register.
00465   *         Reading the contents of the CID
register in SPI mode is a simple
00466   *         read-block transaction.
00467   * @param  Cid pointer on an CID register s
tructure
00468   * @retval SD status
00469   */
00470 static uint8_t SD_GetCIDRegister(SD_CID* Cid
)
00471 {
00472   uint32_t counter = 0;
00473   uint8_t rvalue = MSD_ERROR;
00474   uint8_t CID_Tab[16];
00475
00476   /* Send CMD10 (CID register) and Wait for
response in the R1 format (0x00 is no errors) */
00477   if (SD_IO_WriteCmd(SD_CMD_SEND_CID, 0, 0xF
F, SD_RESPONSE_NO_ERROR) == HAL_OK)
00478   {
```

```
00479     if (SD_IO_WaitResponse(SD_START_DATA_SIN
GLE_BLOCK_READ) == HAL_OK)
00480     {
00481       /* Store CID register value on CID_Tab
 */
00482       for (counter = 0; counter < 16; counte
r++)
00483       {
00484         CID_Tab[counter] = SD_IO_ReadByte();
00485       }
00486
00487       /* Get CRC bytes (not really needed by
 us, but required by SD) */
00488       SD_IO_WriteByte(SD_DUMMY_BYTE);
00489       SD_IO_WriteByte(SD_DUMMY_BYTE);
00490
00491       /* Set response value to success */
00492       rvalue = MSD_OK;
00493     }
00494   }
00495
00496   /* Send dummy byte: 8 Clock pulses of dela
y */
00497   SD_IO_WriteDummy();
00498
00499   if(rvalue == MSD_OK)
00500   {
00501     /* Byte 0 */
00502     Cid->ManufacturerID = CID_Tab[0];
00503
00504     /* Byte 1 */
00505     Cid->OEM_AppliID = CID_Tab[1] << 8;
00506
00507     /* Byte 2 */
00508     Cid->OEM_AppliID |= CID_Tab[2];
00509
00510     /* Byte 3 */
```

```
00511     Cid->ProdName1 = CID_Tab[3] << 24;
00512
00513     /* Byte 4 */
00514     Cid->ProdName1 |= CID_Tab[4] << 16;
00515
00516     /* Byte 5 */
00517     Cid->ProdName1 |= CID_Tab[5] << 8;
00518
00519     /* Byte 6 */
00520     Cid->ProdName1 |= CID_Tab[6];
00521
00522     /* Byte 7 */
00523     Cid->ProdName2 = CID_Tab[7];
00524
00525     /* Byte 8 */
00526     Cid->ProdRev = CID_Tab[8];
00527
00528     /* Byte 9 */
00529     Cid->ProdSN = CID_Tab[9] << 24;
00530
00531     /* Byte 10 */
00532     Cid->ProdSN |= CID_Tab[10] << 16;
00533
00534     /* Byte 11 */
00535     Cid->ProdSN |= CID_Tab[11] << 8;
00536
00537     /* Byte 12 */
00538     Cid->ProdSN |= CID_Tab[12];
00539
00540     /* Byte 13 */
00541     Cid->Reserved1 |= (CID_Tab[13] & 0xF0) >
> 4;
00542     Cid->ManufactDate = (CID_Tab[13] & 0x0F)
 << 8;
00543
00544     /* Byte 14 */
00545     Cid->ManufactDate |= CID_Tab[14];
```

```
00546
00547     /* Byte 15 */
00548     Cid->CID_CRC = (CID_Tab[15] & 0xFE) >> 1
;
00549     Cid->Reserved2 = 1;
00550   }
00551   /* Return the reponse */
00552   return rvalue;
00553 }
00554
00555 /**
00556   * @brief  Send 5 bytes command to the SD c
ard and get response
00557   * @param  Cmd The user expected command to
 send to SD card.
00558   * @param  Arg The command argument.
00559   * @param  Crc The CRC.
00560   * @param  Response Expected response from
the SD card
00561   * @retval SD status
00562   */
00563 static uint8_t SD_SendCmd(uint8_t Cmd, uint3
2_t Arg, uint8_t Crc, uint8_t Response)
00564 {
00565   uint8_t status = MSD_ERROR;
00566
00567   if(SD_IO_WriteCmd(Cmd, Arg, Crc, Response)
 == HAL_OK)
00568   {
00569     status = MSD_OK;
00570   }
00571
00572   /* Send Dummy Byte */
00573   SD_IO_WriteDummy();
00574
00575   return status;
00576 }
```

```
00577
00578 /**
00579   * @brief  Get SD card data response.
00580   * @retval The SD status: Read data respons
e xxx0<status>1
00581   *         - status 010: Data accecpted
00582   *         - status 101: Data rejected due
to a crc error
00583   *         - status 110: Data rejected due
to a Write error.
00584   *         - status 111: Data rejected due
to other error.
00585   */
00586 static SD_Info SD_GetDataResponse(void)
00587 {
00588   uint32_t counter = 0;
00589   SD_Info response, rvalue;
00590
00591   while (counter <= 64)
00592   {
00593     /* Read response */
00594     response = (SD_Info)SD_IO_ReadByte();
00595     /* Mask unused bits */
00596     response &= 0x1F;
00597     switch (response)
00598     {
00599       case SD_DATA_OK:
00600       {
00601         rvalue = SD_DATA_OK;
00602         break;
00603       }
00604       case SD_DATA_CRC_ERROR:
00605         return SD_DATA_CRC_ERROR;
00606       case SD_DATA_WRITE_ERROR:
00607         return SD_DATA_WRITE_ERROR;
00608       default:
00609       {
```

```
00610           rvalue = SD_DATA_OTHER_ERROR;
00611           break;
00612         }
00613       }
00614       /* Exit loop in case of data ok */
00615       if (rvalue == SD_DATA_OK)
00616         break;
00617       /* Increment loop counter */
00618       counter++;
00619     }
00620
00621     /* Wait null data */
00622     while (SD_IO_ReadByte() == 0);
00623
00624     /* Return response */
00625     return response;
00626 }
00627
00628 /**
00629   * @brief  Returns the SD status.
00630   * @retval The SD status.
00631   */
00632 uint8_t BSP_SD_GetStatus(void)
00633 {
00634   return MSD_OK;
00635 }
00636
00637 /**
00638   * @brief  Put SD in Idle state.
00639   * @retval SD status
00640   */
00641 static uint8_t SD_GoIdleState(void)
00642 {
00643   /* Send CMD0 (SD_CMD_GO_IDLE_STATE) to put
 SD in SPI mode and
00644      Wait for In Idle State Response (R1 For
mat) equal to 0x01 */
```

```
00645   if (SD_SendCmd(SD_CMD_GO_IDLE_STATE, 0, 0x
95, SD_IN_IDLE_STATE) != MSD_OK)
00646   {
00647     /* No Idle State Response: return respon
se failue */
00648     return MSD_ERROR;
00649   }
00650
00651   /*----------Activates the card initializat
ion process-----------*/
00652   /* Send CMD1 (Activates the card process)
until response equal to 0x0 and
00653      Wait for no error Response (R1 Format)
equal to 0x00 */
00654   while (SD_SendCmd(SD_CMD_SEND_OP_COND, 0,
0xFF, SD_RESPONSE_NO_ERROR) != MSD_OK);
00655
00656   return MSD_OK;
00657 }
00658 /**
00659   * @brief  Erases the specified memory area
 of the given SD card.
00660   * @param  StartAddr Start byte address
00661   * @param  EndAddr End byte address
00662   * @retval SD status
00663   */
00664 uint8_t BSP_SD_Erase(uint32_t StartAddr, uin
t32_t EndAddr)
00665 {
00666   uint8_t rvalue = MSD_ERROR;
00667
00668   /* Send CMD32 (Erase group start) and chec
k if the SD acknowledged the erase command: R1 res
ponse (0x00: no errors) */
00669   if (SD_SendCmd(SD_CMD_SD_ERASE_GRP_START,
StartAddr, 0xFF, SD_RESPONSE_NO_ERROR) == MSD_OK)
00670   {
```

```
00671     /* Send CMD33 (Erase group end) and Chec
k if the SD acknowledged the erase command: R1 res
ponse (0x00: no errors) */
00672     if (SD_SendCmd(SD_CMD_SD_ERASE_GRP_END,
EndAddr, 0xFF, SD_RESPONSE_NO_ERROR) == MSD_OK)
00673     {
00674       /* Send CMD38 (Erase) and Check if the
 SD acknowledged the erase command: R1 response (0
x00: no errors) */
00675       if (SD_SendCmd(SD_CMD_ERASE, 0, 0xFF,
SD_RESPONSE_NO_ERROR) == MSD_OK)
00676       {
00677         /* Verify that SD card is ready to u
se after the specific command ERASE */
00678         if (SD_IO_WaitResponse(SD_RESPONSE_N
O_ERROR) == HAL_OK)
00679           rvalue = MSD_OK;
00680       }
00681     }
00682   }
00683
00684   /* Return the reponse */
00685   return rvalue;
00686 }
00687 /**
00688   * @}
00689   */
00690
00691 /**
00692   * @}
00693   */
00694
00695 /**
00696   * @}
00697   */
00698
00699 /**
```

```
00700   * @}
00701   */
00702
00703 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32303C_EVAL BSP User Manual

Private Constants

STM32303C-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	**EEPROM_CMD_WREN** 0x06
#define	**EEPROM_CMD_WRDI** 0x04
#define	**EEPROM_CMD_RDSR** 0x05
#define	**EEPROM_CMD_WRSR** 0x01
#define	**EEPROM_CMD_WRITE** 0x02
#define	**EEPROM_CMD_READ** 0x03
#define	**EEPROM_WIP_FLAG** 0x01
#define	**__STM32303C_EVAL_BSP_VERSION_MAIN** (0x02) STM32303C EVAL BSP Driver version number V2.1.2.
#define	**__STM32303C_EVAL_BSP_VERSION_SUB1** (0x01)
#define	**__STM32303C_EVAL_BSP_VERSION_SUB2** (0x02)
#define	**__STM32303C_EVAL_BSP_VERSION_RC** (0x00)
#define	**__STM32303C_EVAL_BSP_VERSION**

Define Documentation

#define __STM32303C_EVAL_BSP_VERSION

Value:

```
((__STM32303C_EVAL_BSP_VERSION_MAIN << 24)\
                                                |(
__STM32303C_EVAL_BSP_VERSION_SUB1 << 16)\
                                                |(
__STM32303C_EVAL_BSP_VERSION_SUB2 << 8 )\
                                                |(
__STM32303C_EVAL_BSP_VERSION_RC))
```

Definition at line **87** of file **stm32303c_eval.c**.

Referenced by **BSP_GetVersion()**.

#define __STM32303C_EVAL_BSP_VERSION_MAIN (0x02)

STM32303C EVAL BSP Driver version number V2.1.2.

[31:24] main version

Definition at line **83** of file **stm32303c_eval.c**.

#define __STM32303C_EVAL_BSP_VERSION_RC (0x00)

[7:0] release candidate

Definition at line **86** of file **stm32303c_eval.c**.

#define __STM32303C_EVAL_BSP_VERSION_SUB1 (0x01)

[23:16] sub1 version

Definition at line **84** of file **stm32303c_eval.c**.

#define __STM32303C_EVAL_BSP_VERSION_SUB2 (0x02)

[15:8] sub2 version

Definition at line **85** of file **stm32303c_eval.c**.

#define EEPROM_CMD_RDSR 0x05

Read Status Register instruction

Definition at line **73** of file **stm32303c_eval.c**.

Referenced by **EEPROM_SPI_IO_WaitEepromStandbyState()**.

#define EEPROM_CMD_READ 0x03

Read from Memory instruction

Definition at line **76** of file **stm32303c_eval.c**.

Referenced by **EEPROM_SPI_IO_ReadData()**.

#define EEPROM_CMD_WRDI 0x04

Write disable instruction

Definition at line **72** of file **stm32303c_eval.c**.

Referenced by **EEPROM_SPI_IO_WriteData()**.

#define EEPROM_CMD_WREN 0x06

Write enable instruction

Definition at line 71 of file stm32303c_eval.c.

Referenced by EEPROM_SPI_IO_WriteData().

#define EEPROM_CMD_WRITE 0x02

Write to Memory instruction

Definition at line 75 of file stm32303c_eval.c.

Referenced by EEPROM_SPI_IO_WriteData().

#define EEPROM_CMD_WRSR 0x01

Write Status Register instruction

Definition at line 74 of file stm32303c_eval.c.

#define EEPROM_WIP_FLAG 0x01

Write In Progress (WIP) flag

Definition at line 78 of file stm32303c_eval.c.

Referenced by EEPROM_SPI_IO_WaitEepromStandbyState().

#define LCD_READ_REG 0x03

Definition at line 64 of file stm32303c_eval.c.

Referenced by LCD_IO_ReadData().

#define LCD_WRITE_REG 0x02

Definition at line 63 of file stm32303c_eval.c.

Referenced by LCD_IO_WriteMultipleData().

#define READ_STATUS 0x01

Definition at line 62 of file stm32303c_eval.c.

#define SD_DUMMY_BYTE 0xFF

Definition at line 67 of file stm32303c_eval.c.

Referenced by SD_IO_Init(), and SD_IO_WriteDummy().

#define SD_NO_RESPONSE_EXPECTED 0x80

Definition at line 68 of file stm32303c_eval.c.

Referenced by SD_IO_WriteCmd().

#define SET_INDEX 0x00

Definition at line 61 of file stm32303c_eval.c.

Referenced by LCD_IO_WriteReg().

#define START_BYTE 0x70

Definition at line 60 of file stm32303c_eval.c.

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

STM32303C_EVAL BSP User Manual

Private Macros

STM32303C-EVAL LCD

Defines

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

Define Documentation

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

Definition at line **101** of file **stm32303c_eval_lcd.c**.

Referenced by **BSP_LCD_DrawLine()**.

STM32303C_EVAL BSP User Manual

Enumerations

Exported Types

STM32303C_EVAL AUDIO

Enumerations

enum	**AUDIO_StatusTypeDef** { **AUDIO_OK** = 0x00, **AUDIO_ERROR** = 0x01, **AUDIO_TIMEOUT** = 0x02 }

Enumeration Type Documentation

enum AUDIO_StatusTypeDef

Enumerator:

AUDIO_OK

AUDIO_ERROR

AUDIO_TIMEOUT

Definition at line **64** of file **stm32303c_eval_audio.h**.

STM32303C_EVAL BSP User Manual

Exported Constants

STM32303C_EVAL AUDIO

Defines

#define	**AUDIO_I2C_ADDRESS** 0x94
#define	**I2Sx** SPI3
#define	**I2Sx_CLK_ENABLE**() __HAL_RCC_SPI3_CLK_ENABLE()
#define	**I2Sx_CLK_DISABLE**() __HAL_RCC_SPI3_CLK_DISABLE()
#define	**I2Sx_FORCE_RESET**() __HAL_RCC_SPI3_FORCE_RESET
#define	**I2Sx_RELEASE_RESET**() __HAL_RCC_SPI3_RELEASE_R
#define	**I2Sx_WS_PIN** GPIO_PIN_4
#define	**I2Sx_MCK_PIN** GPIO_PIN_9
#define	**I2Sx_SCK_PIN** GPIO_PIN_10
#define	**I2Sx_DIN_PIN** GPIO_PIN_12
#define	**I2Sx_WS_GPIO_PORT** GPIOA
#define	**I2Sx_MCK_GPIO_PORT** GPIOA
#define	**I2Sx_SCK_DIN_GPIO_PORT** GPIOC
#define	**I2Sx_MCK_WS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIO
#define	**I2Sx_MCK_WS_GPIO_CLK_DISABLE**() __HAL_RCC_GPIC
#define	**I2Sx_WS_AF** GPIO_AF6_SPI3
#define	**I2Sx_MCK_AF** GPIO_AF5_SPI3
#define	**I2Sx_SCK_DIN_GPIO_CLK_ENABLE**() __HAL_RCC_GPIO
#define	**I2Sx_SCK_DIN_GPIO_CLK_DISABLE**() __HAL_RCC_GPIC
#define	**I2Sx_SCK_DIN_AF** GPIO_AF6_SPI3
#define	**I2Sx_DMAx_CLK_ENABLE**() __HAL_RCC_DMA2_CLK_EN
#define	**I2Sx_DMAx_CLK_DISABLE**() __HAL_RCC_DMA2_CLK_DI
#define	**I2Sx_DMAx_CHANNEL** DMA2_Channel2
#define	**I2Sx_DMAx_IRQ** DMA2_Channel2_IRQn
#define	**I2Sx_DMAx_PERIPH_DATA_SIZE** DMA_PDATAALIGN_HAL
#define	**I2Sx_DMAx_MEM_DATA_SIZE** DMA_MDATAALIGN_HALFV
#define	**DMA_MAX_SZE** 0xFFFF
#define	**AUDIO_OUT_IRQ_PREPRIO** ((uint32_t)0x0E) /* Select the p level(0 is the highest) */
#define	**AUDIO_OUT_IRQ_SUBPRIO** 0 /* Select the sub-priority leve
#define	**AUDIODATA_SIZE** 2 /* 16-bits audio data size */

```
#define  DMA_MAX(_X_)  (((_X_) <= DMA_MAX_SZE)? (_X_):DMA_M
```

Define Documentation

#define AUDIO_I2C_ADDRESS 0x94

Definition at line 81 of file stm32303c_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 ((uint32_t)0x0E) /* Select the

Definition at line 120 of file stm32303c_eval_audio.h.

Referenced by I2Sx_MspInit().

#define AUDIO_OUT_IRQ_SUBPRIO 0 /* Select the sub-priority lev

Definition at line 121 of file stm32303c_eval_audio.h.

Referenced by I2Sx_MspInit().

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

Definition at line 127 of file stm32303c_eval_audio.h.

#define DMA_MAX (_X_) (((_X_) <= DMA_MAX_SZE)? (_X_):DMA

Definition at line 129 of file stm32303c_eval_audio.h.

Referenced by **BSP_AUDIO_OUT_Play()**.

#define DMA_MAX_SZE 0xFFFF

Definition at line **117** of file **stm32303c_eval_audio.h**.

#define I2Sx SPI3

Definition at line **88** of file **stm32303c_eval_audio.h**.

Referenced by **HAL_I2S_ErrorCallback()**, **HAL_I2S_TxCpltCallback()**, **HAL_I2S_TxHalfCpltCallback()**, and **I2Sx_Init()**.

#define I2Sx_CLK_DISABLE ( ) __HAL_RCC_SPI3_CLK_DISABLE

Definition at line **90** of file **stm32303c_eval_audio.h**.

#define I2Sx_CLK_ENABLE ( ) __HAL_RCC_SPI3_CLK_ENABLE(

Definition at line **89** of file **stm32303c_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

#define I2Sx_DIN_PIN GPIO_PIN_12

Definition at line **97** of file **stm32303c_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

#define I2Sx_DMAx_CHANNEL DMA2_Channel2

Definition at line 113 of file stm32303c_eval_audio.h.

Referenced by I2Sx_MspInit().

#define I2Sx_DMAx_CLK_DISABLE ( ) __HAL_RCC_DMA2_CLK_

Definition at line 112 of file stm32303c_eval_audio.h.

#define I2Sx_DMAx_CLK_ENABLE ( ) __HAL_RCC_DMA2_CLK_E

Definition at line 111 of file stm32303c_eval_audio.h.

Referenced by I2Sx_MspInit().

#define I2Sx_DMAx_IRQ DMA2_Channel2_IRQn

Definition at line 114 of file stm32303c_eval_audio.h.

Referenced by I2Sx_MspInit().

#define I2Sx_DMAx_MEM_DATA_SIZE DMA_MDATAALIGN_HALF

Definition at line 116 of file stm32303c_eval_audio.h.

Referenced by I2Sx_MspInit().

#define I2Sx_DMAx_PERIPH_DATA_SIZE DMA_PDATAALIGN_HA

Definition at line 115 of file stm32303c_eval_audio.h.

Referenced by I2Sx_MspInit().

#define I2Sx_FORCE_RESET ( ) __HAL_RCC_SPI3_FORCE_RESI

Definition at line 91 of file stm32303c_eval_audio.h.

Referenced by I2Sx_MspInit().

#define I2Sx_MCK_AF GPIO_AF5_SPI3

Definition at line 105 of file stm32303c_eval_audio.h.

Referenced by I2Sx_MspInit().

#define I2Sx_MCK_GPIO_PORT GPIOA

Definition at line 100 of file stm32303c_eval_audio.h.

Referenced by I2Sx_MspInit().

#define I2Sx_MCK_PIN GPIO_PIN_9

Definition at line 95 of file stm32303c_eval_audio.h.

Referenced by I2Sx_MspInit().

#define I2Sx_MCK_WS_GPIO_CLK_DISABLE ( ) __HAL_RCC_GP

Definition at line 103 of file stm32303c_eval_audio.h.

#define I2Sx_MCK_WS_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPI

Definition at line 102 of file stm32303c_eval_audio.h.

Referenced by **I2Sx_MspInit()**.

#define I2Sx_RELEASE_RESET ( ) __HAL_RCC_SPI3_RELEASE_

Definition at line **92** of file **stm32303c_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

#define I2Sx_SCK_DIN_AF GPIO_AF6_SPI3

Definition at line **108** of file **stm32303c_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

#define I2Sx_SCK_DIN_GPIO_CLK_DISABLE ( ) __HAL_RCC_GP

Definition at line **107** of file **stm32303c_eval_audio.h**.

#define I2Sx_SCK_DIN_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPI

Definition at line **106** of file **stm32303c_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

#define I2Sx_SCK_DIN_GPIO_PORT GPIOC

Definition at line **101** of file **stm32303c_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

#define I2Sx_SCK_PIN GPIO_PIN_10

Definition at line **96** of file **stm32303c_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

#define I2Sx_WS_AF GPIO_AF6_SPI3

Definition at line **104** of file **stm32303c_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

#define I2Sx_WS_GPIO_PORT GPIOA

Definition at line **99** of file **stm32303c_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

#define I2Sx_WS_PIN GPIO_PIN_4

Definition at line **94** of file **stm32303c_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

STM32303C_EVAL BSP User Manual

Functions

Link Operation functions

STM32303C-EVAL Common

Functions

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_SPI_IO_Init** (void) Initializes the EEPROM SPI and put it into StandBy State (Ready for data transfer).
void	**EEPROM_SPI_IO_WriteByte** (uint8_t Data) Write a byte on the EEPROM.

uint8_t	**EEPROM_SPI_IO_ReadByte** (void) Read a byte from the EEPROM.
HAL_StatusTypeDef	**EEPROM_SPI_IO_WriteData** (uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Write data to SPI EEPROM driver.
HAL_StatusTypeDef	**EEPROM_SPI_IO_ReadData** (uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Read data from SPI EEPROM driver.
HAL_StatusTypeDef	**EEPROM_SPI_IO_WaitEepromStandbyState** (void) Wait response from the SPI EEPROM.
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 peripherals used by the Audio Codec driver.
void	**AUDIO_IO_DeInit** (void) DeInitializes Audio low level.
void	**AUDIO_IO_Write** (uint16_t DevAddress, uint8_t Reg, uint8_t Value) Writes a single data on the Audio Codec.
uint8_t	**AUDIO_IO_Read** (uint16_t DevAddress, uint8_t Reg) Reads a single data from the Audio Codec.
void	**AUDIO_IO_Delay** (uint32_t Delay) Wait for loop in ms.

Function Documentation

void AUDIO_IO_DeInit (void)

DeInitializes Audio low level.

Note:
This function is intentionally kept empty, user should define it.

Definition at line **1452** of file **stm32303c_eval.c**.

void AUDIO_IO_Delay (uint32_t Delay)

Wait for loop in ms.

Parameters:
Delay in ms.

Return values:
None

Definition at line **1489** of file **stm32303c_eval.c**.

void AUDIO_IO_Init (void)

Initializes peripherals used by the Audio Codec driver.

Return values:
None

Definition at line **1443** of file **stm32303c_eval.c**.

References **I2Cx_Init()**.

uint8_t AUDIO_IO_Read (uint16_t DevAddress,
uint8_t Reg
)

Reads a single data from the Audio Codec.

Parameters:

DevAddress	Target device address
Reg	Target Register address

Return values:

Data	to be read

Definition at line **1475** of file **stm32303c_eval.c**.

References **I2Cx_ReadData()**.

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

Writes a single data on the Audio Codec.

Parameters:

DevAddress	Target device address
Reg	Target Register address
Value	Data to be written

Return values:

None	

Definition at line **1464** of file **stm32303c_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 **1347** of file **stm32303c_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 **1385** of file **stm32303c_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 **1373** of file **stm32303c_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 **1360** of file **stm32303c_eval.c**.

References **I2Cx_WriteBuffer()**.

Referenced by **EEPROM_I2C_WritePage()**.

void EEPROM_SPI_IO_Init (void)

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

Return values:

None

Definition at line **1152** of file **stm32303c_eval.c**.

References **EEPROM_CS_GPIO_CLK_ENABLE**, **EEPROM_CS_GPIO_PORT**, **EEPROM_CS_HIGH**, **EEPROM_CS_PIN**, and **SPIx_Init()**.

Referenced by **EEPROM_SPI_Init()**.

uint8_t EEPROM_SPI_IO_ReadByte (void)

Read a byte from the EEPROM.

Return values:

uint8_t (The received byte).

Definition at line **1189** of file **stm32303c_eval.c**.

References **SPIx_Read()**.

HAL_StatusTypeDef EEPROM_SPI_IO_ReadData (uint16_t MemAd
uint8_t * pBuffe
uint32_t BufferS
)

Read data from SPI EEPROM driver.

Parameters:

MemAddress	Internal memory address
pBuffer	Pointer to data buffer
BufferSize	Amount of data to be read

Return values:

HAL_StatusTypeDef HAL Status

< Select the EEPROM: Chip Select low

< Send "Write to Memory " instruction

< Send MemAddress high nibble address byte to write to

< Send WriteAddr medium nibble address byte to write to

< Send WriteAddr low nibble address byte to write to

< while there is data to be read

< Read a byte from the EEPROM

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

Definition at line **1269** of file **stm32303c_eval.c**.

References **EEPROM_CMD_READ**, **EEPROM_CS_HIGH**, **EEPROM_CS_LOW**, **SPIx_Read()**, and **SPIx_Write()**.

Referenced by **EEPROM_SPI_ReadBuffer()**.

HAL_StatusTypeDef EEPROM_SPI_IO_WaitEepromStandbyState (v

Wait response from the SPI EEPROM.

Return values:

HAL_StatusTypeDef HAL Status

< Select the EEPROM: Chip Select low

< Send "Read Status Register" instruction

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

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

< Deselect the EEPROM: Chip Select high

Definition at line **1304** of file **stm32303c_eval.c**.

References **EEPROM_CMD_RDSR**, **EEPROM_CS_HIGH**, **EEPROM_CS_LOW**, **EEPROM_WIP_FLAG**, **SPIx_Read()**, and **SPIx_Write()**.

Referenced by **EEPROM_SPI_IO_WriteData()**, and **EEPROM_SPI_WaitEepromStandbyState()**.

void EEPROM_SPI_IO_WriteByte (uint8_t Data)

Write a byte on the EEPROM.

Parameters:

Data byte to send.

Return values:

None

Definition at line **1179** of file **stm32303c_eval.c**.

References **SPIx_Write()**.

HAL_StatusTypeDef EEPROM_SPI_IO_WriteData (uint16_t MemAd
uint8_t * pBuffe
uint32_t BufferS
)

Write data to SPI EEPROM driver.

Parameters:

MemAddress	Internal memory address
pBuffer	Pointer to data buffer
BufferSize	Amount of data to be read

Return values:

HAL_StatusTypeDef HAL Status

< Enable the write access to the EEPROM

< Select the EEPROM: Chip Select low

< Send "Write Enable" instruction

< Deselect the EEPROM: Chip Select high

< Select the EEPROM: Chip Select low

< Send "Write to Memory " instruction

< Send MemAddress high nibble address byte to write to

< Send MemAddress medium nibble address byte to write to

< Send MemAddress low nibble address byte to write to

< while there is data to be written on the EEPROM

< Send the current byte

< **Point** on the next byte to be written

< Wait the end of EEPROM writing

< Disable the write access to the EEROM

< Send "Write Disable" instruction

< Deselect the EEPROM: Chip Select high

Definition at line **1207** of file **stm32303c_eval.c**.

References **EEPROM_CMD_WRDI**, **EEPROM_CMD_WREN**, **EEPROM_CMD_WRITE**, **EEPROM_CS_HIGH**, **EEPROM_CS_LOW**, **EEPROM_SPI_IO_WaitEepromStandbyState()**, and **SPIx_Write()**.

Referenced by **EEPROM_SPI_WritePage()**.

void LCD_Delay (uint32_t Delay)

Wait for loop in ms.

Parameters:

Delay in ms.

Return values:

None

Definition at line **987** of file **stm32303c_eval.c**.

void LCD_IO_Init (void)

Configures the LCD_SPI interface.

Return values:

None

Definition at line **871** of file **stm32303c_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 **947** of file **stm32303c_eval.c**.

References **heval_Spi**, **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 **898** of file **stm32303c_eval.c**.

References **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 **926** of file **stm32303c_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 **999** of file **stm32303c_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 **1056** of file **stm32303c_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 **1110** of file **stm32303c_eval.c**.

References **SD_IO_ReadByte()**.

Referenced by **BSP_SD_Erase()**, **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 **1046** of file **stm32303c_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 **1075** of file **stm32303c_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 **1136** of file **stm32303c_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 **1395** of file **stm32303c_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 **1432** of file **stm32303c_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 **1421** of file **stm32303c_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 **1408** of file **stm32303c_eval.c**.

References **I2Cx_WriteBuffer()**.

STM32303C_EVAL BSP User Manual

Private Variables

STM32303C-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 115 of file stm32303c_eval_lcd.c.

Referenced by LCD_DrawChar().

LCD_DrawPropTypeDef DrawProp

Definition at line 110 of file stm32303c_eval_lcd.c.

LCD_DrvTypeDef* lcd_drv [static]

Definition at line 112 of file stm32303c_eval_lcd.c.

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

STM32303C_EVAL BSP User Manual

Exported Functions

STM32303C_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.
uint8_t	**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).
uint8_t	**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) Audio OUT Error callback function.

Function Documentation

uint8_t 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:

AUDIO_OK if correct communication, else wrong communication

Definition at line **247** of file **stm32303c_eval_audio.c**.

References **hAudioOutI2s**.

void BSP_AUDIO_OUT_Error_CallBack (void)

Audio OUT Error callback function.

Return values:

None

Definition at line **462** of file **stm32303c_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 **454** of file **stm32303c_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:

AUDIO_OK	if correct communication, else wrong communication

Definition at line **182** of file **stm32303c_eval_audio.c**.

References **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, **I2Sx_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 **259** of file **stm32303c_eval_audio.c**.

References **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **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 **227** of file **stm32303c_eval_audio.c**.

References **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **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 **280** of file **stm32303c_eval_audio.c**.

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

uint8_t BSP_AUDIO_OUT_SetFrequency (uint32_t AudioFreq)

Update the audio frequency.

Parameters:

AudioFreq	Audio frequency used to play the audio stream.

Return values:

AUDIO_OK	if correct communication, else wrong communication

Definition at line **392** of file **stm32303c_eval_audio.c**.

References **I2Sx_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 **351** of file **stm32303c_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 **373** of file **stm32303c_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 **331** of file **stm32303c_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_SW: for software power off (by writing registers). Then no need to reconfigure the Codec after power on.

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

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **303** of file **stm32303c_eval_audio.c**.

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

void BSP_AUDIO_OUT_TransferComplete_CallBack (void)

Manages the DMA full Transfer complete event.

Return values:

None

Definition at line **446** of file **stm32303c_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 **432** of file **stm32303c_eval_audio.c**.

References **BSP_AUDIO_OUT_Error_CallBack()**, and **I2Sx**.

void HAL_I2S_TxCpltCallback (I2S_HandleTypeDef * hi2s)

Tx Transfer completed callbacks.

Parameters:

hi2s I2S handle

Return values:

None

Definition at line **403** of file **stm32303c_eval_audio.c**.

References **BSP_AUDIO_OUT_TransferComplete_CallBack()**, and **I2Sx**.

void HAL_I2S_TxHalfCpltCallback (I2S_HandleTypeDef * hi2s)

Tx Transfer Half completed callbacks.

Parameters:

hi2s I2S handle

Return values:

None

Definition at line **417** of file **stm32303c_eval_audio.c**.

References **BSP_AUDIO_OUT_HalfTransfer_CallBack()**, and **I2Sx**.

STM32303C_EVAL BSP User Manual

Exported Functions

STM32303C-EVAL Common

Functions

uint32_t	**BSP_GetVersion** (void) This method returns the STM32303C EVAL BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LED GPIO.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** Button_Mode) Configures push button GPIO and EXTI Line.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
uint8_t	**BSP_JOY_Init** (**JOYMode_TypeDef** Joy_Mode) Configures all button of the joystick in GPIO or EXTI modes.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the current joystick status.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) Configures COM port.

Function Documentation

void BSP_COM_Init (COM_TypeDef COM,
UART_HandleTypeDef * huart
)

Configures COM port.

Parameters:

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

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

Return values:

None

Definition at line **496** of file **stm32303c_eval.c**.

References **COM_RX_AF**, **COM_RX_PIN**, **COM_RX_PORT**, **COM_TX_AF**, **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 STM32303C EVAL BSP Driver revision.

Return values:

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

Definition at line **270** of file **stm32303c_eval.c**.

References __STM32303C_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
- JOY_NONE

Definition at line 469 of file stm32303c_eval.c.

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

uint8_t BSP_JOY_Init (JOYMode_TypeDef Joy_Mode)

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

Parameters:

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

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

Return values:

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

Definition at line **420** of file **stm32303c_eval.c**.

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

void BSP_LED_Init (Led_TypeDef Led)

Configures LED GPIO.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Return values:

None

Definition at line **285** of file **stm32303c_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 **328** of file **stm32303c_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 **313** of file **stm32303c_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 **343** of file **stm32303c_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_KEY: Key Push Button

Return values:

The Button GPIO pin value

Definition at line **406** of file **stm32303c_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_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 **360** of file **stm32303c_eval.c**.

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

STM32303C_EVAL BSP User Manual

Exported Functions

STM32303C-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.

void	**EEPROM_SPI_IO_Init** (void) Initializes the EEPROM SPI and put it into StandBy State (Ready for data transfer).
void	**EEPROM_SPI_IO_WriteByte** (uint8_t Data) Write a byte on the EEPROM.
uint8_t	**EEPROM_SPI_IO_ReadByte** (void) Read a byte from the EEPROM.
HAL_StatusTypeDef	**EEPROM_SPI_IO_WriteData** (uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Write data to SPI EEPROM driver.
HAL_StatusTypeDef	**EEPROM_SPI_IO_ReadData** (uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Read data from SPI EEPROM driver.
HAL_StatusTypeDef	**EEPROM_SPI_IO_WaitEepromStandbyState** (void) Wait response from the SPI EEPROM.
void	**EEPROM_SPI_IO_WriteDummy** (void)

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 **193** of file **stm32303c_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 **252** of file **stm32303c_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_M24LR64
- BSP_EEPROM_M24M01
- BSP_EEPROM_M95M01

Return values:

EEPROM_OK (0) if operation is correctly performed, else return value different from EEPROM_OK (0)

Definition at line **216** of file **stm32303c_eval_eeprom.c**.

References **BSP_EEPROM_M24LR64**, **BSP_EEPROM_M24M01**, **BSP_EEPROM_M95M01**, **EEPROM_I2C_Drv**, and **EEPROM_SPI_Drv**.

void BSP_EEPROM_TIMEOUT_UserCallback (void)

Basic management of the timeout situation.

Return values:

None.

Definition at line **426** of file **stm32303c_eval_eeprom.c**.

Referenced by **EEPROM_I2C_WaitEepromStandbyState()**, and **EEPROM_SPI_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 **273** of file **stm32303c_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 **1347** of file **stm32303c_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 **1385** of file **stm32303c_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 **1373** of file **stm32303c_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 **1360** of file **stm32303c_eval.c**.

References **I2Cx_WriteBuffer()**.

Referenced by **EEPROM_I2C_WritePage()**.

void EEPROM_SPI_IO_Init (void)

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

Return values:

None	

Definition at line **1152** of file **stm32303c_eval.c**.

References **EEPROM_CS_GPIO_CLK_ENABLE**, **EEPROM_CS_GPIO_PORT**, **EEPROM_CS_HIGH**, **EEPROM_CS_PIN**, and **SPIx_Init()**.

Referenced by **EEPROM_SPI_Init()**.

uint8_t EEPROM_SPI_IO_ReadByte (void)

Read a byte from the EEPROM.

Return values:

uint8_t (The received byte).

Definition at line **1189** of file **stm32303c_eval.c**.

References **SPIx_Read()**.

HAL_StatusTypeDef EEPROM_SPI_IO_ReadData (uint16_t MemAd
uint8_t * pBuffe
uint32_t BufferS
)

Read data from SPI EEPROM driver.

Parameters:

MemAddress	Internal memory address
pBuffer	Pointer to data buffer
BufferSize	Amount of data to be read

Return values:

HAL_StatusTypeDef HAL Status

< Select the EEPROM: Chip Select low

< Send "Write to Memory " instruction

< Send MemAddress high nibble address byte to write to

< Send WriteAddr medium nibble address byte to write to

< Send WriteAddr low nibble address byte to write to

< while there is data to be read

< Read a byte from the EEPROM

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

Definition at line **1269** of file **stm32303c_eval.c**.

References **EEPROM_CMD_READ**, **EEPROM_CS_HIGH**, **EEPROM_CS_LOW**, **SPIx_Read()**, and **SPIx_Write()**.

Referenced by **EEPROM_SPI_ReadBuffer()**.

HAL_StatusTypeDef EEPROM_SPI_IO_WaitEepromStandbyState (v

Wait response from the SPI EEPROM.

Return values:

HAL_StatusTypeDef HAL Status

< Select the EEPROM: Chip Select low

< Send "Read Status Register" instruction

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

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

< Deselect the EEPROM: Chip Select high

Definition at line **1304** of file **stm32303c_eval.c**.

References **EEPROM_CMD_RDSR**, **EEPROM_CS_HIGH**, **EEPROM_CS_LOW**, **EEPROM_WIP_FLAG**, **SPIx_Read()**, and **SPIx_Write()**.

Referenced by **EEPROM_SPI_IO_WriteData()**, and **EEPROM_SPI_WaitEepromStandbyState()**.

void EEPROM_SPI_IO_WriteByte (uint8_t Data)

Write a byte on the EEPROM.

Parameters:

Data byte to send.

Return values:

None

Definition at line **1179** of file **stm32303c_eval.c**.

References **SPIx_Write()**.

HAL_StatusTypeDef EEPROM_SPI_IO_WriteData (uint16_t MemAd
uint8_t * pBuffe
uint32_t BufferS
)

Write data to SPI EEPROM driver.

Parameters:

MemAddress	Internal memory address
pBuffer	Pointer to data buffer
BufferSize	Amount of data to be read

Return values:

HAL_StatusTypeDef	HAL Status

< Enable the write access to the EEPROM

< Select the EEPROM: Chip Select low

< Send "Write Enable" instruction

< Deselect the EEPROM: Chip Select high

< Select the EEPROM: Chip Select low

< Send "Write to Memory " instruction

< Send MemAddress high nibble address byte to write to

< Send MemAddress medium nibble address byte to write to

< Send MemAddress low nibble address byte to write to

< while there is data to be written on the EEPROM

< Send the current byte

< **Point** on the next byte to be written

< Wait the end of EEPROM writing

< Disable the write access to the EEROM

< Send "Write Disable" instruction

< Deselect the EEPROM: Chip Select high

Definition at line **1207** of file **stm32303c_eval.c**.

References **EEPROM_CMD_WRDI**, **EEPROM_CMD_WREN**, **EEPROM_CMD_WRITE**, **EEPROM_CS_HIGH**, **EEPROM_CS_LOW**, **EEPROM_SPI_IO_WaitEepromStandbyState()**, and **SPIx_Write()**.

Referenced by **EEPROM_SPI_WritePage()**.

void EEPROM_SPI_IO_WriteDummy (void)

STM32303C_EVAL BSP User Manual

Defines

Exported Constants

STM32303C-EVAL EEPROM

Defines

#define	**EEPROM_ADDRESS_M24M01** 0xA4 /* EEPROM M24M01-HR used */
#define	**EEPROM_ADDRESS_M24LR64_A01** 0xA0 /* RF EEPROM ANT7-M24LR-A01 used */
#define	**EEPROM_ADDRESS_M24LR64_A02** 0xA6 /* RF EEPROM ANT7-M24LR-A02 used */
#define	**EEPROM_PAGESIZE_M24M01** 28 /* EEPROM M24M01-HR used */
#define	**EEPROM_PAGESIZE_M24LR64** 4 /* RF EEPROM ANT7-M24LR-A used */
#define	**EEPROM_PAGESIZE_M95M01** 256 /* EEPROM M95M01 used */
#define	**EEPROM_OK** 0
#define	**EEPROM_FAIL** 1
#define	**EEPROM_TIMEOUT** 2
#define	**BSP_EEPROM_M24LR64** 1 /* RF I2C EEPROM M24LR64 */
#define	**BSP_EEPROM_M24M01** 2 /* I2C EEPROM M24M01 */
#define	**BSP_EEPROM_M95M01** 3 /* SPI EEPROM M95M01 */
#define	**EEPROM_MAX_TRIALS** 300

Define Documentation

#define BSP_EEPROM_M24LR64 1 /* RF I2C EEPROM M24LR64 */

Definition at line 98 of file stm32303c_eval_eeprom.h.

Referenced by BSP_EEPROM_SelectDevice().

#define BSP_EEPROM_M24M01 2 /* I2C EEPROM M24M01 */

Definition at line 99 of file stm32303c_eval_eeprom.h.

Referenced by BSP_EEPROM_SelectDevice().

#define BSP_EEPROM_M95M01 3 /* SPI EEPROM M95M01 */

Definition at line 100 of file stm32303c_eval_eeprom.h.

Referenced by BSP_EEPROM_SelectDevice().

#define EEPROM_ADDRESS_M24LR64_A01 0xA0 /* RF EEPROM

Definition at line 85 of file stm32303c_eval_eeprom.h.

Referenced by EEPROM_I2C_Init().

#define EEPROM_ADDRESS_M24LR64_A02 0xA6 /* RF EEPROM

Definition at line 86 of file stm32303c_eval_eeprom.h.

Referenced by EEPROM_I2C_Init().

#define EEPROM_ADDRESS_M24M01 0xA4 /* EEPROM M24M01-H

Definition at line 84 of file stm32303c_eval_eeprom.h.

Referenced by EEPROM_I2C_Init().

#define EEPROM_FAIL 1

Definition at line 94 of file stm32303c_eval_eeprom.h.

Referenced by BSP_EEPROM_Init(), BSP_EEPROM_ReadBuffer(), BSP_EEPROM_WriteBuffer(), EEPROM_I2C_Init(), EEPROM_I2C_ReadBuffer(), EEPROM_I2C_WritePage(), EEPROM_SPI_Init(), EEPROM_SPI_ReadBuffer(), and EEPROM_SPI_WritePage().

#define EEPROM_MAX_TRIALS 300

Definition at line 103 of file stm32303c_eval_eeprom.h.

Referenced by EEPROM_I2C_Init(), and EEPROM_I2C_WaitEepromStandbyState().

#define EEPROM_OK 0

Definition at line 93 of file stm32303c_eval_eeprom.h.

Referenced by BSP_EEPROM_WriteBuffer(), EEPROM_I2C_Init(), EEPROM_I2C_ReadBuffer(), EEPROM_I2C_WaitEepromStandbyState(), EEPROM_I2C_WritePage(), EEPROM_SPI_Init(), EEPROM_SPI_ReadBuffer(), EEPROM_SPI_WaitEepromStandbyState(), and EEPROM_SPI_WritePage().

#define EEPROM_PAGESIZE_M24LR64 4 /* RF EEPROM ANT7-M2

Definition at line 89 of file stm32303c_eval_eeprom.h.

Referenced by EEPROM_I2C_Init().

#define EEPROM_PAGESIZE_M24M01 28 /* EEPROM M24M01-HR

Definition at line 88 of file stm32303c_eval_eeprom.h.

Referenced by EEPROM_I2C_Init().

#define EEPROM_PAGESIZE_M95M01 256 /* EEPROM M95M01 us

Definition at line 90 of file stm32303c_eval_eeprom.h.

Referenced by EEPROM_SPI_Init().

#define EEPROM_TIMEOUT 2

Definition at line 95 of file stm32303c_eval_eeprom.h.

Referenced by EEPROM_I2C_WaitEepromStandbyState(), and EEPROM_SPI_WaitEepromStandbyState().

STM32303C_EVAL BSP User Manual

Exported Functions

STM32303C-EVAL LCD

Functions

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

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

void	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 **252** of file **stm32303c_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 **277** of file **stm32303c_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 297 of file stm32303c_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 802 of file stm32303c_eval_lcd.c.

References **lcd_drv**.

void BSP_LCD_DisplayOn (void)

Enables the display.

Return values:

None

Definition at line **793** of file **stm32303c_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 315 of file stm32303c_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 374 of file stm32303c_eval_lcd.c.

References BSP_LCD_DisplayStringAt(), and LEFT_MODE.

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

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

Parameters:

Xpos	Bmp X position in the LCD
Ypos	Bmp Y position in the LCD
pBmp	Pointer to Bmp picture address in the internal Flash

Return values:

None

Definition at line **660** of file **stm32303c_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 **548** of file **stm32303c_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 **627** of file **stm32303c_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 **404** of file **stm32303c_eval_lcd.c**.

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

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

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

Draws an uni-line (between two points).

Parameters:

X1	**Point** 1 X position
Y1	**Point** 1 Y position
X2	**Point** 2 X position
Y2	**Point** 2 Y position

Return values:

None

Definition at line **455** of file **stm32303c_eval_lcd.c**.

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

Referenced by **BSP_LCD_DrawPolygon()**.

void BSP_LCD_DrawPolygon (pPoint pPoints,
uint16_t PointCount
)

Draws an poly-line (between many points).

Parameters:

pPoints	Pointer to the points array
PointCount	Number of points

Return values:

None

Definition at line **598** of file **stm32303c_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 **530** of file **stm32303c_eval_lcd.c**.

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

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

Draws a vertical line.

Parameters:

Xpos X position

Ypos Y position

Length Line length

Return values:

None

Definition at line **428** of file **stm32303c_eval_lcd.c**.

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

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

void BSP_LCD_FillCircle (uint16_t Xpos,
uint16_t Ypos,

uint16_t Radius
)

Draws a full circle.

Parameters:

Xpos	X position
Ypos	Y position
Radius	Circle radius

Return values:

None

Definition at line **713** of file **stm32303c_eval_lcd.c**.

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

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

Draws a full ellipse.

Parameters:

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

Return values:

None

Definition at line **763** of file **stm32303c_eval_lcd.c**.

References **BSP_LCD_DrawVLine()**.

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 **696** of file **stm32303c_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 **203** of file **stm32303c_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**.

sFONT * BSP_LCD_GetFont (void)

Gets the LCD text font.

Return values:

Used font

Definition at line **242** of file **stm32303c_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 **194** of file **stm32303c_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 **176** of file **stm32303c_eval_lcd.c**.

References **lcd_drv**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_ClearStringLine()**, **BSP_LCD_DisplayStringAt()**, **BSP_LCD_DrawBitmap()**, and **BSP_LCD_DrawVLine()**.

uint32_t BSP_LCD_GetYSize (void)

Gets the LCD Y size.

Return values:

Used LCD Y size

Definition at line **185** of file **stm32303c_eval_lcd.c**.

References **lcd_drv**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_DrawBitmap()**, and **BSP_LCD_DrawVLine()**.

uint8_t BSP_LCD_Init (void)

Initializes the LCD.

Return values:

LCD state

Definition at line **139** of file **stm32303c_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 385 of file stm32303c_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 223 of file stm32303c_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 **233** of file **stm32303c_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 **213** of file **stm32303c_eval_lcd.c**.

References **LCD_DrawPropTypeDef::TextColor**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_ClearStringLine()**, **BSP_LCD_FillCircle()**, and **BSP_LCD_FillRect()**.

Generated on Wed May 31 2017 10:00:44 for STM32303C_EVAL BSP
User Manual by doxygen 1.7.6.1

STM32303C_EVAL BSP User Manual

Exported Functions

STM32303C-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_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.

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 **664** of file **stm32303c_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, **SD_CMD_ERASE**, **SD_CMD_SD_ERASE_GRP_END**, **SD_CMD_SD_ERASE_GRP_START**, **SD_IO_WaitResponse()**, **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 **197** of file **stm32303c_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 **632** of file **stm32303c_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 **152** of file **stm32303c_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 **176** of file **stm32303c_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 **220** of file **stm32303c_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 **286** of file **stm32303c_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**.

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 **999** of file **stm32303c_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 **1056** of file **stm32303c_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 **1110** of file **stm32303c_eval.c**.

References **SD_IO_ReadByte()**.

Referenced by **BSP_SD_Erase()**, **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 **1046** of file **stm32303c_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 **1075** of file **stm32303c_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 **1136** of file **stm32303c_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()**.

STM32303C_EVAL BSP User Manual

Private Functions

STM32303C-EVAL SD

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.

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 **664** of file **stm32303c_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, **SD_CMD_ERASE**, **SD_CMD_SD_ERASE_GRP_END**, **SD_CMD_SD_ERASE_GRP_START**, **SD_IO_WaitResponse()**, **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 **197** of file **stm32303c_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 **632** of file **stm32303c_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 **152** of file **stm32303c_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 **176** of file **stm32303c_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 **220** of file **stm32303c_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 286 of file stm32303c_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 **470** of file **stm32303c_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 **352** of file **stm32303c_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 **586** of file **stm32303c_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 **641** of file **stm32303c_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()**.

```
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 **563** of file **stm32303c_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, **SD_IO_WriteCmd()**, and **SD_IO_WriteDummy()**.

Referenced by **BSP_SD_Erase()**, and **SD_GoIdleState()**.

STM32303C_EVAL BSP User Manual

Exported Functions

STM32303C-EVAL TSENSOR

Functions

uint32_t	**BSP_TSENSOR_Init** (void) Initializes peripherals used by the I2C Temperature Sensor driver.
uint8_t	**BSP_TSENSOR_ReadStatus** (void) Returns the Temperature Sensor status.
uint16_t	**BSP_TSENSOR_ReadTemp** (void) Read Temperature register of TS751.

Function Documentation

uint32_t BSP_TSENSOR_Init (void)

Initializes peripherals used by the I2C Temperature Sensor driver.

Return values:

TSENSOR status

Definition at line **91** of file **stm32303c_eval_tsensor.c**.

References **tsensor_drv**, **TSENSOR_ERROR**, **TSENSOR_I2C_ADDRESS**, **TSENSOR_MAX_TRIALS**, and **TSENSOR_OK**.

uint8_t BSP_TSENSOR_ReadStatus (void)

Returns the Temperature Sensor status.

Return values:

The Temperature Sensor status.

Definition at line **128** of file **stm32303c_eval_tsensor.c**.

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

uint16_t BSP_TSENSOR_ReadTemp (void)

Read Temperature register of TS751.

Return values:

STTS751 measured temperature value.

Definition at line **137** of file **stm32303c_eval_tsensor.c**.

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

STM32303C_EVAL BSP User Manual

Private Functions

STM32303C-EVAL TSENSOR

Functions

uint32_t	**BSP_TSENSOR_Init** (void) Initializes peripherals used by the I2C Temperature Sensor driver.
uint8_t	**BSP_TSENSOR_ReadStatus** (void) Returns the Temperature Sensor status.
uint16_t	**BSP_TSENSOR_ReadTemp** (void) Read Temperature register of TS751.

Function Documentation

uint32_t BSP_TSENSOR_Init (void)

Initializes peripherals used by the I2C Temperature Sensor driver.

Return values:

TSENSOR status

Definition at line **91** of file **stm32303c_eval_tsensor.c**.

References **tsensor_drv**, **TSENSOR_ERROR**, **TSENSOR_I2C_ADDRESS**, **TSENSOR_MAX_TRIALS**, and **TSENSOR_OK**.

uint8_t BSP_TSENSOR_ReadStatus (void)

Returns the Temperature Sensor status.

Return values:

The Temperature Sensor status.

Definition at line **128** of file **stm32303c_eval_tsensor.c**.

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

uint16_t BSP_TSENSOR_ReadTemp (void)

Read Temperature register of TS751.

Return values:

STTS751 measured temperature value.

Definition at line **137** of file **stm32303c_eval_tsensor.c**.

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

STM32303C_EVAL BSP User Manual

Enumerations

Exported Types

STM32303C-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_KEY = 0, **BUTTON_SEL** = 1, **BUTTON_LEFT** = 2, **BUTTON_RIGHT** = 3,
	BUTTON_DOWN = 4, **BUTTON_UP** = 5
	}
	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 } COM Types Definition. More...

Enumeration Type Documentation

enum Button_TypeDef

BUTTON Types Definition.

Enumerator:

BUTTON_KEY
BUTTON_SEL
BUTTON_LEFT
BUTTON_RIGHT
BUTTON_DOWN
BUTTON_UP

Definition at line **97** of file **stm32303c_eval.h**.

enum ButtonMode_TypeDef

Enumerator:

BUTTON_MODE_GPIO
BUTTON_MODE_EXTI

Definition at line **107** of file **stm32303c_eval.h**.

enum COM_TypeDef

COM Types Definition.

Enumerator:

COM1

Definition at line **135** of file **stm32303c_eval.h**.

enum JOYMode_TypeDef

Enumerator:

JOY_MODE_GPIO

JOY_MODE_EXTI

Definition at line **126** of file **stm32303c_eval.h**.

enum JOYState_TypeDef

JOYSTICK Types Definition.

Enumerator:

JOY_SEL

JOY_LEFT

JOY_RIGHT

JOY_DOWN

JOY_UP

JOY_NONE

Definition at line **116** of file **stm32303c_eval.h**.

enum Led_TypeDef

LED Types Definition.

Enumerator:

LED1

LED2

LED3

LED4

LED_GREEN

LED_ORANGE
LED_RED
LED_BLUE

Definition at line **81** of file **stm32303c_eval.h**.

STM32303C_EVAL BSP User Manual

Private Variables

STM32303C-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**]
GPIO_TypeDef *	**JOY_PORT** [**JOYn**] JOYSTICK variables.
const uint16_t	**JOY_PIN** [**JOYn**]
const uint8_t	**JOY_IRQn** [**JOYn**]
USART_TypeDef *	**COM_USART** [**COMn**] = {**EVAL_COM1**} 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**}
const uint16_t	**COM_TX_AF** [**COMn**] = {**EVAL_COM1_TX_AF**}
const uint16_t	**COM_RX_AF** [**COMn**] = {**EVAL_COM1_RX_AF**}
uint32_t	**I2cxTimeout** = **EVAL_I2Cx_TIMEOUT_MAX** BUS variables.
I2C_HandleTypeDef	**heval_I2c**
uint32_t	**SpixTimeout** = **EVAL_SPIx_TIMEOUT_MAX**

static SPI_HandleTypeDef **heval_Spi**

Variable Documentation

const uint16_t BUTTON_IRQn[BUTTONn]

Initial value:

```
 {KEY_BUTTON_EXTI_IRQn,
                                                SEL_JOY_EX
TI_IRQn,
                                                LEFT_JOY_E
XTI_IRQn,
                                                RIGHT_JOY_
EXTI_IRQn,
                                                DOWN_JOY_E
XTI_IRQn,
                                                UP_JOY_EXT
I_IRQn}
```

Definition at line **128** of file **stm32303c_eval.c**.

Referenced by **BSP_PB_Init()**.

const uint16_t BUTTON_PIN[BUTTONn]

Initial value:

```
 {KEY_BUTTON_PIN,
                                              SEL_JOY_PIN
,
                                              LEFT_JOY_PIN
,
                                              RIGHT_JOY_P
IN,
                                              DOWN_JOY_PIN
,
                                              UP_JOY_PIN}
```

Definition at line **121** of file **stm32303c_eval.c**.

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

GPIO_TypeDef* BUTTON_PORT[BUTTONn]

Initial value:

```
 {KEY_BUTTON_GPIO_PORT,
                                      SEL_JOY_GPI
O_PORT,
                                      LEFT_JOY_GP
IO_PORT,
                                      RIGHT_JOY_G
PIO_PORT,
                                      DOWN_JOY_GP
IO_PORT,
                                      UP_JOY_GPIO
_PORT}
```

BUTTON variables.

Definition at line **114** of file **stm32303c_eval.c**.

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

const uint16_t COM_RX_AF[COMn] = {EVAL_COM1_RX_AF}

Definition at line **172** of file **stm32303c_eval.c**.

Referenced by **BSP_COM_Init()**.

const uint16_t COM_RX_PIN[COMn] = {EVAL_COM1_RX_PIN}

Definition at line **168** of file **stm32303c_eval.c**.

Referenced by **BSP_COM_Init()**.

GPIO_TypeDef* COM_RX_PORT[COMn] = {EVAL_COM1_RX_GPIO_

Definition at line **164** of file **stm32303c_eval.c**.

Referenced by **BSP_COM_Init()**.

const uint16_t COM_TX_AF[COMn] = {EVAL_COM1_TX_AF}

Definition at line **170** of file **stm32303c_eval.c**.

Referenced by **BSP_COM_Init()**.

const uint16_t COM_TX_PIN[COMn] = {EVAL_COM1_TX_PIN}

Definition at line **166** of file **stm32303c_eval.c**.

Referenced by **BSP_COM_Init()**.

GPIO_TypeDef* COM_TX_PORT[COMn] = {EVAL_COM1_TX_GPIO_

Definition at line **162** of file **stm32303c_eval.c**.

Referenced by **BSP_COM_Init()**.

USART_TypeDef* COM_USART[COMn] = {EVAL_COM1}

COM variables.

Definition at line **160** of file **stm32303c_eval.c**.

Referenced by **BSP_COM_Init()**.

I2C_HandleTypeDef heval_I2c

Definition at line 180 of file stm32303c_eval.c.

Referenced by I2Cx_Error(), I2Cx_Init(), I2Cx_IsDeviceReady(), I2Cx_ReadBuffer(), I2Cx_ReadData(), I2Cx_WriteBuffer(), and I2Cx_WriteData().

SPI_HandleTypeDef heval_Spi `[static]`

Definition at line 185 of file stm32303c_eval.c.

Referenced by LCD_IO_ReadData(), SPIx_Error(), SPIx_Init(), SPIx_Read(), and SPIx_Write().

uint32_t I2cxTimeout = EVAL_I2Cx_TIMEOUT_MAX

BUS variables.

Definition at line 179 of file stm32303c_eval.c.

Referenced by I2Cx_IsDeviceReady(), I2Cx_ReadBuffer(), I2Cx_ReadData(), I2Cx_WriteBuffer(), and I2Cx_WriteData().

const uint8_t JOY_IRQn[JOYn]

Initial value:

```
 {SEL_JOY_EXTI_IRQn,
                                     LEFT_JOY_EXTI_IRQn
,
                                     RIGHT_JOY_EXTI_IR
Qn,
                                     DOWN_JOY_EXTI_IRQn
```

```
,
                                    UP_JOY_EXTI_IRQn}
```

Definition at line **150** of file **stm32303c_eval.c**.

Referenced by **BSP_JOY_Init()**.

const uint16_t JOY_PIN[JOYn]

Initial value:

```
 {SEL_JOY_PIN,
                                    LEFT_JOY_PIN,
                                    RIGHT_JOY_PIN,
                                    DOWN_JOY_PIN,
                                    UP_JOY_PIN}
```

Definition at line **144** of file **stm32303c_eval.c**.

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

GPIO_TypeDef* JOY_PORT[JOYn]

Initial value:

```
 {SEL_JOY_GPIO_PORT,
                                    LEFT_JOY_GPIO_PORT
,
                                    RIGHT_JOY_GPIO_PO
RT,
                                    DOWN_JOY_GPIO_PORT
,
                                    UP_JOY_GPIO_PORT}
```

JOYSTICK variables.

Definition at line **138** of file **stm32303c_eval.c**.

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

const uint16_t LED_PIN[LEDn]

Initial value:

```
{LED1_PIN,
                                    LED2_PIN,
                                    LED3_PIN,
                                    LED4_PIN}
```

Definition at line **106** of file **stm32303c_eval.c**.

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

GPIO_TypeDef* LED_PORT[LEDn]

Initial value:

```
{LED1_GPIO_PORT,
                                    LED2_GPIO_PORT,
                                    LED3_GPIO_PORT,
                                    LED4_GPIO_PORT}
```

LED variables.

Definition at line **101** of file **stm32303c_eval.c**.

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

uint32_t SpixTimeout = EVAL_SPIx_TIMEOUT_MAX

Definition at line **184** of file **stm32303c_eval.c**.

Referenced by **SPIx_Read()**, and **SPIx_Write()**.

STM32303C_EVAL BSP User Manual

Defines

STM32303C-EVAL BUTTON

Exported Constants

Defines

#define	**JOYn** 5
#define	**BUTTONn** 1 + **JOYn**
#define	**KEY_BUTTON_PIN** GPIO_PIN_6 Key push-button.
#define	**KEY_BUTTON_GPIO_PORT** GPIOE
#define	**KEY_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE
#define	**KEY_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_GPIO
#define	**KEY_BUTTON_EXTI_IRQn** EXTI9_5_IRQn
#define	**RIGHT_JOY_PIN** GPIO_PIN_6 Joystick Right push-button.
#define	**RIGHT_JOY_GPIO_PORT** GPIOD
#define	**RIGHT_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOD_
#define	**RIGHT_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOD_
#define	**RIGHT_JOY_EXTI_IRQn** EXTI9_5_IRQn
#define	**LEFT_JOY_PIN** GPIO_PIN_5 Joystick Left push-button.
#define	**LEFT_JOY_GPIO_PORT** GPIOB
#define	**LEFT_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_C
#define	**LEFT_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_C
#define	**LEFT_JOY_EXTI_IRQn** EXTI9_5_IRQn
#define	**UP_JOY_PIN** GPIO_PIN_7 Joystick Up push-button.
#define	**UP_JOY_GPIO_PORT** GPIOE
#define	**UP_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_CLK
#define	**UP_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_CLK
#define	**UP_JOY_EXTI_IRQn** EXTI9_5_IRQn
#define	**DOWN_JOY_PIN** GPIO_PIN_5 Joystick Down push-button.
#define	**DOWN_JOY_GPIO_PORT** GPIOD
#define	**DOWN_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOD_
#define	**DOWN_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOD_

#define	**DOWN_JOY_EXTI_IRQn** EXTI9_5_IRQn
#define	**SEL_JOY_PIN** GPIO_PIN_13 Joystick Sel push-button.
#define	**SEL_JOY_GPIO_PORT** GPIOC
#define	**SEL_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOC_CL
#define	**SEL_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOC_CL
#define	**SEL_JOY_EXTI_IRQn** EXTI15_10_IRQn
#define	**BUTTONx_GPIO_CLK_ENABLE**(__BUTTON__)
#define	**BUTTONx_GPIO_CLK_DISABLE**(__BUTTON__)
#define	**JOYx_GPIO_CLK_ENABLE**(__JOY__)
#define	**JOYx_GPIO_CLK_DISABLE**(__JOY__)

Define Documentation

#define BUTTONn 1 + JOYn

Definition at line 200 of file stm32303c_eval.h.

#define BUTTONx_GPIO_CLK_DISABLE (__BUTTON__)

Value:

```
(((__BUTTON__) == BUTTON_KEY) ? KEY_BUTTON_GPIO_C
LK_DISABLE()  :\

((__BUTTON__) == BUTTON_SEL) ? SEL_JOY_GPIO_CLK_D
ISABLE()  :\

((__BUTTON__) == BUTTON_LEFT) ? LEFT_JOY_GPIO_CLK
_DISABLE()  :\

((__BUTTON__) == BUTTON_RIGHT) ? RIGHT_JOY_GPIO_C
LK_DISABLE()  :\

((__BUTTON__) == BUTTON_DOWN) ? DOWN_JOY_GPIO_CLK
_DISABLE()  :\

((__BUTTON__) == BUTTON_UP) ? UP_JOY_GPIO_CLK_DIS
ABLE() : 0 )
```

Definition at line 263 of file stm32303c_eval.h.

#define BUTTONx_GPIO_CLK_ENABLE (__BUTTON__)

Value:

```
do { if ((__BUTTON__) == BUTTON_KEY) KEY_BUTTON_G
PIO_CLK_ENABLE(); else\
```

```
    if ((__BUTTON__) == BUTTON_SEL) SEL_JOY_GPIO_
CLK_ENABLE(); else\

    if ((__BUTTON__) == BUTTON_LEFT) LEFT_JOY_GPI
O_CLK_ENABLE(); else\

    if ((__BUTTON__) == BUTTON_RIGHT) RIGHT_JOY_G
PIO_CLK_ENABLE(); else\

    if ((__BUTTON__) == BUTTON_DOWN) DOWN_JOY_GPI
O_CLK_ENABLE(); else\

    if ((__BUTTON__) == BUTTON_UP) UP_JOY_GPIO_CL
K_ENABLE();} while(0)
```

Definition at line **256** of file **stm32303c_eval.h**.

Referenced by **BSP_PB_Init()**.

#define DOWN_JOY_EXTI_IRQn EXTI9_5_IRQn

Definition at line **245** of file **stm32303c_eval.h**.

#define DOWN_JOY_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOI

Definition at line **244** of file **stm32303c_eval.h**.

#define DOWN_JOY_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOD

Definition at line **243** of file **stm32303c_eval.h**.

#define DOWN_JOY_GPIO_PORT GPIOD

Definition at line **242** of file **stm32303c_eval.h**.

#define DOWN_JOY_PIN GPIO_PIN_5

Joystick Down push-button.

Definition at line **241** of file **stm32303c_eval.h**.

#define JOYn 5

Definition at line **199** of file **stm32303c_eval.h**.

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

#define JOYx_GPIO_CLK_DISABLE (__JOY__)

Value:

```
(((__JOY__) == JOY_SEL) ? SEL_JOY_GPIO_CLK_DISABLE
() :\
                                                  (
(__JOY__) == JOY_LEFT) ? LEFT_JOY_GPIO_CLK_DISABLE
() :\
                                                  (
(__JOY__) == JOY_RIGHT) ? RIGHT_JOY_GPIO_CLK_DISA
BLE() :\
                                                  (
(__JOY__) == JOY_DOWN) ? DOWN_JOY_GPIO_CLK_DISABLE
() :\
                                                  (
(__JOY__) == JOY_UP) ? UP_JOY_GPIO_CLK_DISABLE()
: 0 )
```

Definition at line **276** of file **stm32303c_eval.h**.

#define JOYx_GPIO_CLK_ENABLE (__JOY__)

Value:

```
do { if ((__JOY__) == JOY_SEL) SEL_JOY_GPIO_CLK_E
NABLE(); else\

   if ((__JOY__) == JOY_LEFT) LEFT_JOY_GPIO_CLK_E
NABLE(); else\

   if ((__JOY__) == JOY_RIGHT) RIGHT_JOY_GPIO_CLK
_ENABLE(); else\

   if ((__JOY__) == JOY_DOWN) DOWN_JOY_GPIO_CLK_E
NABLE(); else\

   if ((__JOY__) == JOY_UP) UP_JOY_GPIO_CLK_ENABLE
();} while(0)
```

Definition at line 270 of file stm32303c_eval.h.

Referenced by BSP_JOY_Init().

#define KEY_BUTTON_EXTI_IRQn EXTI9_5_IRQn

Definition at line 209 of file stm32303c_eval.h.

#define KEY_BUTTON_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPI

Definition at line 208 of file stm32303c_eval.h.

#define KEY_BUTTON_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPI(

Definition at line 207 of file stm32303c_eval.h.

#define KEY_BUTTON_GPIO_PORT GPIOE

Definition at line 206 of file stm32303c_eval.h.

#define KEY_BUTTON_PIN GPIO_PIN_6

Key push-button.

Definition at line 205 of file stm32303c_eval.h.

#define LEFT_JOY_EXTI_IRQn EXTI9_5_IRQn

Definition at line 227 of file stm32303c_eval.h.

#define LEFT_JOY_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOB_

Definition at line 226 of file stm32303c_eval.h.

#define LEFT_JOY_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOB_

Definition at line 225 of file stm32303c_eval.h.

#define LEFT_JOY_GPIO_PORT GPIOB

Definition at line 224 of file stm32303c_eval.h.

#define LEFT_JOY_PIN GPIO_PIN_5

Joystick Left push-button.

Definition at line 223 of file stm32303c_eval.h.

#define RIGHT_JOY_EXTI_IRQn EXTI9_5_IRQn

Definition at line 218 of file stm32303c_eval.h.

#define RIGHT_JOY_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOD

Definition at line 217 of file stm32303c_eval.h.

#define RIGHT_JOY_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOD

Definition at line 216 of file stm32303c_eval.h.

#define RIGHT_JOY_GPIO_PORT GPIOD

Definition at line 215 of file stm32303c_eval.h.

#define RIGHT_JOY_PIN GPIO_PIN_6

Joystick Right push-button.

Definition at line 214 of file stm32303c_eval.h.

#define SEL_JOY_EXTI_IRQn EXTI15_10_IRQn

Definition at line 254 of file stm32303c_eval.h.

#define SEL_JOY_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOC_

Definition at line **253** of file **stm32303c_eval.h**.

#define SEL_JOY_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOC_C

Definition at line **252** of file **stm32303c_eval.h**.

#define SEL_JOY_GPIO_PORT GPIOC

Definition at line **251** of file **stm32303c_eval.h**.

#define SEL_JOY_PIN GPIO_PIN_13

Joystick Sel push-button.

Definition at line **250** of file **stm32303c_eval.h**.

#define UP_JOY_EXTI_IRQn EXTI9_5_IRQn

Definition at line **236** of file **stm32303c_eval.h**.

#define UP_JOY_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOE_C

Definition at line **235** of file **stm32303c_eval.h**.

#define UP_JOY_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOE_CL

Definition at line **234** of file **stm32303c_eval.h**.

#define UP_JOY_GPIO_PORT GPIOE

Definition at line **233** of file **stm32303c_eval.h**.

#define UP_JOY_PIN GPIO_PIN_7

Joystick Up push-button.

Definition at line **232** of file **stm32303c_eval.h**.

STM32303C_EVAL BSP User Manual

STM32303C-EVAL COM

Exported Constants

Defines

#define	**COMn** 1
#define	**EVAL_COM1** USART1 Definition for COM port1, connected to USART1.
#define	**EVAL_COM1_CLK_ENABLE**() __HAL_RCC_USART1_CLK_
#define	**EVAL_COM1_CLK_DISABLE**() __HAL_RCC_USART1_CLK
#define	**EVAL_COM1_TX_PIN** GPIO_PIN_4
#define	**EVAL_COM1_TX_GPIO_PORT** GPIOC
#define	**EVAL_COM1_TX_GPIO_CLK_ENABLE**() __HAL_RCC_GPI
#define	**EVAL_COM1_TX_GPIO_CLK_DISABLE**() __HAL_RCC_GP
#define	**EVAL_COM1_TX_AF** GPIO_AF7_USART1
#define	**EVAL_COM1_RX_PIN** GPIO_PIN_1
#define	**EVAL_COM1_RX_GPIO_PORT** GPIOE
#define	**EVAL_COM1_RX_GPIO_CLK_ENABLE**() __HAL_RCC_GPI
#define	**EVAL_COM1_RX_GPIO_CLK_DISABLE**() __HAL_RCC_GP
#define	**EVAL_COM1_RX_AF** GPIO_AF7_USART1
#define	**EVAL_COM1_IRQn** USART1_IRQn
#define	**COMx_CLK_ENABLE**(__INDEX__) do { if ((__INDEX__) == **EVAL_COM1_CLK_ENABLE**();} while(0)
#define	**COMx_CLK_DISABLE**(__INDEX__) (((__INDEX__) == **COM** **EVAL_COM1_CLK_DISABLE**() : 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	**EVAL_I2Cx_SCL_PIN** GPIO_PIN_6
#define	**EVAL_I2Cx_SCL_GPIO_PORT** GPIOF
#define	**EVAL_I2Cx_SDA_PIN** GPIO_PIN_10
#define	**EVAL_I2Cx_SDA_GPIO_PORT** GPIOA

#define **EVAL_I2Cx_SCL_SDA_AF** GPIO_AF4_I2C2

#define **EVAL_I2Cx** I2C2

#define **EVAL_I2Cx_CLK_ENABLE**() __HAL_RCC_I2C2_CLK_ENA

#define **EVAL_I2Cx_SDA_GPIO_CLK_ENABLE**() __HAL_RCC_GPI

#define **EVAL_I2Cx_SCL_GPIO_CLK_ENABLE**() __HAL_RCC_GPI

#define **EVAL_I2Cx_FORCE_RESET**() __HAL_RCC_I2C2_FORCE_

#define **EVAL_I2Cx_RELEASE_RESET**() __HAL_RCC_I2C2_RELE

#define **EVAL_I2Cx_EV_IRQn** I2C2_EV_IRQn

#define **EVAL_I2Cx_EV_IRQHandler** I2C2_EV_IRQHandler

#define **EVAL_I2Cx_ER_IRQn** I2C2_ER_IRQn

#define **EVAL_I2Cx_ER_IRQHandler** I2C2_ER_IRQHandler

#define **EVAL_I2Cx_TIMEOUT_MAX** 1000

#define **EVAL_SPIx** SPI2

Definition for SPI Interface pins (SPI2 used)

#define **EVAL_SPIx_CLK_ENABLE**() __HAL_RCC_SPI2_CLK_ENA

#define **EVAL_SPIx_CLK_DISABLE**() __HAL_RCC_SPI2_CLK_DIS

#define **EVAL_SPIx_FORCE_RESET**() __HAL_RCC_SPI2_FORCE_

#define **EVAL_SPIx_RELEASE_RESET**() __HAL_RCC_SPI2_RELE

#define **EVAL_SPIx_SCK_AF** GPIO_AF5_SPI2

#define **EVAL_SPIx_SCK_GPIO_PORT** GPIOF

#define **EVAL_SPIx_SCK_PIN** GPIO_PIN_9

#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_AF** GPIO_AF5_SPI2

#define **EVAL_SPIx_MISO_MOSI_GPIO_PORT** GPIOB

#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_14

#define **EVAL_SPIx_MOSI_PIN** GPIO_PIN_15

#define **EVAL_SPIx_TIMEOUT_MAX** 1000

Define Documentation

#define COMn 1

Definition at line 291 of file stm32303c_eval.h.

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

Definition at line 315 of file stm32303c_eval.h.

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

Definition at line 314 of file stm32303c_eval.h.

Referenced by BSP_COM_Init().

#define COMx_RX_GPIO_CLK_DISABLE (__INDEX__) (((__INDE

Definition at line 321 of file stm32303c_eval.h.

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

Definition at line 320 of file stm32303c_eval.h.

Referenced by BSP_COM_Init().

#define COMx_TX_GPIO_CLK_DISABLE (__INDEX__) (((__INDE

Definition at line 318 of file stm32303c_eval.h.

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

Definition at line 317 of file stm32303c_eval.h.

Referenced by BSP_COM_Init().

#define EVAL_COM1 USART1

Definition for COM port1, connected to USART1.

Definition at line 296 of file stm32303c_eval.h.

#define EVAL_COM1_CLK_DISABLE ( ) __HAL_RCC_USART1_CL

Definition at line 298 of file stm32303c_eval.h.

#define EVAL_COM1_CLK_ENABLE ( ) __HAL_RCC_USART1_CL

Definition at line 297 of file stm32303c_eval.h.

#define EVAL_COM1_IRQn USART1_IRQn

Definition at line 312 of file stm32303c_eval.h.

#define EVAL_COM1_RX_AF GPIO_AF7_USART1

Definition at line 310 of file stm32303c_eval.h.

#define EVAL_COM1_RX_GPIO_CLK_DISABLE ( ) __HAL_RCC_C

Definition at line 309 of file stm32303c_eval.h.

#define EVAL_COM1_RX_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line 308 of file stm32303c_eval.h.

#define EVAL_COM1_RX_GPIO_PORT GPIOE

Definition at line 307 of file stm32303c_eval.h.

#define EVAL_COM1_RX_PIN GPIO_PIN_1

Definition at line 306 of file stm32303c_eval.h.

#define EVAL_COM1_TX_AF GPIO_AF7_USART1

Definition at line 304 of file stm32303c_eval.h.

#define EVAL_COM1_TX_GPIO_CLK_DISABLE ( ) __HAL_RCC_G

Definition at line 303 of file stm32303c_eval.h.

#define EVAL_COM1_TX_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line 302 of file stm32303c_eval.h.

#define EVAL_COM1_TX_GPIO_PORT GPIOC

Definition at line 301 of file stm32303c_eval.h.

#define EVAL_COM1_TX_PIN GPIO_PIN_4

Definition at line 300 of file stm32303c_eval.h.

#define EVAL_I2Cx I2C2

Definition at line 337 of file stm32303c_eval.h.

Referenced by I2Cx_Init(), and I2Cx_MspInit().

#define EVAL_I2Cx_CLK_ENABLE ( ) __HAL_RCC_I2C2_CLK_EN

Definition at line 338 of file stm32303c_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_ER_IRQHandler I2C2_ER_IRQHandler

Definition at line 349 of file stm32303c_eval.h.

#define EVAL_I2Cx_ER_IRQn I2C2_ER_IRQn

Definition at line 348 of file stm32303c_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_EV_IRQHandler I2C2_EV_IRQHandler

Definition at line 347 of file stm32303c_eval.h.

#define EVAL_I2Cx_EV_IRQn I2C2_EV_IRQn

Definition at line **346** of file **stm32303c_eval.h**.

Referenced by **I2Cx_MspInit()**.

#define EVAL_I2Cx_FORCE_RESET ( ) __HAL_RCC_I2C2_FORCE

Definition at line **342** of file **stm32303c_eval.h**.

Referenced by **I2Cx_MspInit()**.

#define EVAL_I2Cx_RELEASE_RESET ( ) __HAL_RCC_I2C2_REL

Definition at line **343** of file **stm32303c_eval.h**.

Referenced by **I2Cx_MspInit()**.

#define EVAL_I2Cx_SCL_GPIO_CLK_ENABLE ( ) __HAL_RCC_GI

Definition at line **340** of file **stm32303c_eval.h**.

Referenced by **I2Cx_MspInit()**.

#define EVAL_I2Cx_SCL_GPIO_PORT GPIOF

Definition at line **331** of file **stm32303c_eval.h**.

Referenced by **I2Cx_MspInit()**.

#define EVAL_I2Cx_SCL_PIN GPIO_PIN_6

Definition at line **330** of file **stm32303c_eval.h**.

Referenced by **I2Cx_MspInit()**.

#define EVAL_I2Cx_SCL_SDA_AF GPIO_AF4_I2C2

Definition at line 334 of file stm32303c_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SDA_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line 339 of file stm32303c_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SDA_GPIO_PORT GPIOA

Definition at line 333 of file stm32303c_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SDA_PIN GPIO_PIN_10

Definition at line 332 of file stm32303c_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_TIMEOUT_MAX 1000

Definition at line 363 of file stm32303c_eval.h.

#define EVAL_SPIx SPI2

Definition for SPI Interface pins (SPI2 used)

Definition at line **371** of file **stm32303c_eval.h**.

Referenced by **SPIx_Init()**.

#define EVAL_SPIx_CLK_DISABLE ( ) __HAL_RCC_SPI2_CLK_DI

Definition at line **373** of file **stm32303c_eval.h**.

#define EVAL_SPIx_CLK_ENABLE ( ) __HAL_RCC_SPI2_CLK_EN

Definition at line **372** of file **stm32303c_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_FORCE_RESET ( ) __HAL_RCC_SPI2_FORCE

Definition at line **374** of file **stm32303c_eval.h**.

#define EVAL_SPIx_MISO_MOSI_AF GPIO_AF5_SPI2

Definition at line **383** of file **stm32303c_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_MISO_MOSI_GPIO_CLK_DISABLE ( ) __HAL_

Definition at line **386** of file **stm32303c_eval.h**.

#define EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE ( ) __HAL_

Definition at line 385 of file stm32303c_eval.h.

Referenced by SPIx_MspInit().

#define EVAL_SPIx_MISO_MOSI_GPIO_PORT GPIOB

Definition at line 384 of file stm32303c_eval.h.

Referenced by SPIx_MspInit().

#define EVAL_SPIx_MISO_PIN GPIO_PIN_14

Definition at line 387 of file stm32303c_eval.h.

Referenced by SPIx_MspInit().

#define EVAL_SPIx_MOSI_PIN GPIO_PIN_15

Definition at line 388 of file stm32303c_eval.h.

Referenced by SPIx_MspInit().

#define EVAL_SPIx_RELEASE_RESET ( ) __HAL_RCC_SPI2_REL

Definition at line 375 of file stm32303c_eval.h.

#define EVAL_SPIx_SCK_AF GPIO_AF5_SPI2

Definition at line 377 of file stm32303c_eval.h.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_SCK_GPIO_CLK_DISABLE ( ) __HAL_RCC_G

Definition at line **381** of file **stm32303c_eval.h**.

#define EVAL_SPIx_SCK_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line **380** of file **stm32303c_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_SCK_GPIO_PORT GPIOF

Definition at line **378** of file **stm32303c_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_SCK_PIN GPIO_PIN_9

Definition at line **379** of file **stm32303c_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_TIMEOUT_MAX 1000

Definition at line **395** of file **stm32303c_eval.h**.

STM32303C_EVAL BSP User Manual

Defines

STM32303C-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_0 /* PE.00 */ LCD Control Interface pins.
#define	**LCD_NCS_GPIO_PORT** GPIOE
#define	**LCD_NCS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_CL
#define	**LCD_NCS_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_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_15 /* PE.15 */ SD card Control pins.
#define	**SD_CS_GPIO_PORT** GPIOE
#define	**SD_CS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_CLK_
#define	**SD_CS_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_CLK_
#define	**SD_DETECT_PIN** GPIO_PIN_6 /* PC.06 */ SD Detect Interface pins.
#define	**SD_DETECT_GPIO_PORT** GPIOC
#define	**SD_DETECT_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOC_
#define	**SD_DETECT_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOC_
#define	**SD_DETECT_EXTI_IRQn** EXTI9_5_IRQn
#define	**EEPROM_CS_LOW**() HAL_GPIO_WritePin(**EEPROM_CS_G** **EEPROM_CS_PIN**, GPIO_PIN_RESET)
#define	**EEPROM_CS_HIGH**() HAL_GPIO_WritePin(**EEPROM_CS_C** **EEPROM_CS_PIN**, GPIO_PIN_SET)
#define	**EEPROM_CS_PIN** GPIO_PIN_7 EEPROM Control Interface pins.
#define	**EEPROM_CS_GPIO_PORT** GPIOD

```
#define EEPROM_CS_GPIO_CLK_ENABLE()  __HAL_RCC_GPIOD
#define EEPROM_CS_GPIO_CLK_DISABLE()  __HAL_RCC_GPIOD
```

Define Documentation

#define EEPROM_CS_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIC

Definition at line 452 of file stm32303c_eval.h.

#define EEPROM_CS_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIO

Definition at line 451 of file stm32303c_eval.h.

Referenced by EEPROM_SPI_IO_Init().

#define EEPROM_CS_GPIO_PORT GPIOD

Definition at line 450 of file stm32303c_eval.h.

Referenced by EEPROM_SPI_IO_Init().

#define EEPROM_CS_HIGH ( ) HAL_GPIO_WritePin(EEPROM_CS

Definition at line 444 of file stm32303c_eval.h.

Referenced by EEPROM_SPI_IO_Init(), EEPROM_SPI_IO_ReadData(), EEPROM_SPI_IO_WaitEepromStandbyState(), and EEPROM_SPI_IO_WriteData().

#define EEPROM_CS_LOW ( ) HAL_GPIO_WritePin(EEPROM_CS

Definition at line 443 of file stm32303c_eval.h.

Referenced by EEPROM_SPI_IO_ReadData(),

EEPROM_SPI_IO_WaitEepromStandbyState(), and **EEPROM_SPI_IO_WriteData()**.

#define EEPROM_CS_PIN GPIO_PIN_7

EEPROM Control Interface pins.

Definition at line **449** of file **stm32303c_eval.h**.

Referenced by **EEPROM_SPI_IO_Init()**.

#define LCD_CS_HIGH ( ) HAL_GPIO_WritePin(LCD_NCS_GPIO_

Definition at line **409** of file **stm32303c_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 **408** of file **stm32303c_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_GPIOE_

Definition at line **417** of file **stm32303c_eval.h**.

#define LCD_NCS_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOE_(

Definition at line **416** of file **stm32303c_eval.h**.

Referenced by LCD_IO_Init().

#define LCD_NCS_GPIO_PORT GPIOE

Definition at line 415 of file stm32303c_eval.h.

Referenced by LCD_IO_Init().

#define LCD_NCS_PIN GPIO_PIN_0 /* PE.00 */

LCD Control Interface pins.

Definition at line 414 of file stm32303c_eval.h.

Referenced by LCD_IO_Init().

#define SD_CS_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOE_CL

Definition at line 430 of file stm32303c_eval.h.

#define SD_CS_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOE_CLK

Definition at line 429 of file stm32303c_eval.h.

Referenced by SD_IO_Init().

#define SD_CS_GPIO_PORT GPIOE

Definition at line 428 of file stm32303c_eval.h.

Referenced by SD_IO_Init().

#define SD_CS_HIGH ( ) HAL_GPIO_WritePin(SD_CS_GPIO_POR

Definition at line 422 of file stm32303c_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 421 of file stm32303c_eval.h.

Referenced by SD_IO_WriteCmd().

#define SD_CS_PIN GPIO_PIN_15 /* PE.15 */

SD card Control pins.

Definition at line 427 of file stm32303c_eval.h.

Referenced by SD_IO_Init().

#define SD_DETECT_EXTI_IRQn EXTI9_5_IRQn

Definition at line 439 of file stm32303c_eval.h.

Referenced by SD_IO_Init().

#define SD_DETECT_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIO

Definition at line 438 of file stm32303c_eval.h.

#define SD_DETECT_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOC

Definition at line **437** of file **stm32303c_eval.h**.

Referenced by **SD_IO_Init()**.

#define SD_DETECT_GPIO_PORT GPIOC

Definition at line **436** of file **stm32303c_eval.h**.

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

#define SD_DETECT_PIN GPIO_PIN_6 /* PC.06 */

SD Detect Interface pins.

Definition at line **435** of file **stm32303c_eval.h**.

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

STM32303C_EVAL BSP User Manual

Private Types

STM32303C-EVAL EEPROM

Variables

EEPROM_DrvTypeDef	EEPROM_I2C_Drv
EEPROM_DrvTypeDef	EEPROM_SPI_Drv

Variable Documentation

EEPROM_DrvTypeDef EEPROM_I2C_Drv

Initial value:

```
{
  EEPROM_I2C_Init,
  EEPROM_I2C_ReadBuffer,
  EEPROM_I2C_WritePage
}
```

Definition at line **166** of file **stm32303c_eval_eeprom.c**.

Referenced by **BSP_EEPROM_SelectDevice()**.

EEPROM_DrvTypeDef EEPROM_SPI_Drv

Initial value:

```
{
  EEPROM_SPI_Init,
  EEPROM_SPI_ReadBuffer,
  EEPROM_SPI_WritePage
}
```

Definition at line **174** of file **stm32303c_eval_eeprom.c**.

Referenced by **BSP_EEPROM_SelectDevice()**.

STM32303C_EVAL BSP User Manual

STM32303C_EVAL_EEPROM_Private_Functions

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.
static uint32_t	**EEPROM_SPI_Init** (void) Initializes peripherals used by the SPI EEPROM driver.
static uint32_t	**EEPROM_SPI_ReadBuffer** (uint8_t *pBuffer, uint16_t ReadAddr, uint32_t *NumByteToRead) Reads a block of data from the SPI EEPROM.
static uint32_t	**EEPROM_SPI_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_SPI_WaitEepromStandbyState** (void) Wait for EEPROM SPI Standby state.

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 M24LR64 (A01 & A02). Then try to connect on 1st one (EEPROM_I2C_ADDRESS_A01) and if problem, check the 2nd one (EEPROM_I2C_ADDRESS_A02)

Return values:

EEPROM_OK	(0) if operation is correctly performed, else return value different from EEPROM_OK (0)

Definition at line **445** of file **stm32303c_eval_eeprom.c**.

References **EEPROM_ADDRESS_M24LR64_A01**, **EEPROM_ADDRESS_M24LR64_A02**, **EEPROM_ADDRESS_M24M01**, **EEPROM_FAIL**, **EEPROM_I2C_IO_Init()**, **EEPROM_I2C_IO_IsDeviceReady()**, **EEPROM_MAX_TRIALS**, **EEPROM_OK**, **EEPROM_PAGESIZE_M24LR64**, **EEPROM_PAGESIZE_M24M01**, **EEPROMAddress**, and **EEPROMPageSize**.

static uint32_t EEPROM_I2C_ReadBuffer (uint8_t * pBuffer,
uint16_t ReadAddr,
uint32_t * NumByteToR
) [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 **483** of file **stm32303c_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 **551** of file **stm32303c_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 **520** of file **stm32303c_eval_eeprom.c**.

References **EEPROM_FAIL**, **EEPROM_I2C_IO_WriteData()**, **EEPROM_I2C_WaitEepromStandbyState()**, **EEPROM_OK**, and **EEPROMAddress**.

static uint32_t EEPROM_SPI_Init (void) [static]

Initializes peripherals used by the SPI EEPROM driver.

Return values:

EEPROM_OK (0) if operation is correctly performed, else return value different from EEPROM_OK (0)

Definition at line **568** of file **stm32303c_eval_eeprom.c**.

References **EEPROM_FAIL**, **EEPROM_OK**, **EEPROM_PAGESIZE_M95M01**, **EEPROM_SPI_IO_Init()**, **EEPROM_SPI_WaitEepromStandbyState()**, and **EEPROMPageSize**.

static uint32_t EEPROM_SPI_ReadBuffer (uint8_t * pBuffer, uint16_t ReadAddr, uint32_t * NumByteToR) [static]

Reads a block of data from the SPI EEPROM.

Parameters:

pBuffer pointer to the buffer that receives the data read from the EEPROM.

ReadAddr	EEPROM's internal address to start reading from.
NumByteToRead	pointer to the variable holding number of bytes to be read from the EEPROM.

Note:
The variable pointed by NumByteToRead is reset to 0 when all the data are read from the EEPROM. Application should monitor this variable in order know when the transfer is complete.

Return values:

EEPROM_OK	(0) if operation is correctly performed, else return value different from EEPROM_OK (0) or the timeout user callback.

Definition at line **596** of file **stm32303c_eval_eeprom.c**.

References **EEPROM_FAIL**, **EEPROM_OK**, **EEPROM_SPI_IO_ReadData()**, and **EEPROM_SPI_WaitEepromStandbyState()**.

static uint32_t EEPROM_SPI_WaitEepromStandbyState (void) [st

Wait for EEPROM SPI Standby state.

Note:
This function allows to wait and check that EEPROM has finished the last operation. It is mostly used after Write operation.

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 **661** of file **stm32303c_eval_eeprom.c**.

References **BSP_EEPROM_TIMEOUT_UserCallback()**, **EEPROM_OK**, **EEPROM_SPI_IO_WaitEepromStandbyState()**, and **EEPROM_TIMEOUT**.

Referenced by **EEPROM_SPI_Init()**, **EEPROM_SPI_ReadBuffer()**, and **EEPROM_SPI_WritePage()**.

---

static uint32_t EEPROM_SPI_WritePage (uint8_t * pBuffer,
uint16_t WriteAddr,
uint32_t * NumByteToWr
) [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 **634** of file **stm32303c_eval_eeprom.c**.

References **EEPROM_FAIL**, **EEPROM_OK**, **EEPROM_SPI_IO_WriteData()**, and **EEPROM_SPI_WaitEepromStandbyState()**.

STM32303C_EVAL BSP User Manual

Private Variables

STM32303C-EVAL EEPROM

Variables

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

Variable Documentation

EEPROM_DrvTypeDef* EEPROM_SelectedDevice = 0 [static]

Definition at line 140 of file stm32303c_eval_eeprom.c.

__IO uint16_t EEPROMAddress = 0

Definition at line 135 of file stm32303c_eval_eeprom.c.

Referenced by EEPROM_I2C_Init(), EEPROM_I2C_ReadBuffer(), EEPROM_I2C_WaitEepromStandbyState(), and EEPROM_I2C_WritePage().

__IO uint16_t EEPROMDataRead

Definition at line 137 of file stm32303c_eval_eeprom.c.

__IO uint8_t EEPROMDataWrite

Definition at line 138 of file stm32303c_eval_eeprom.c.

__IO uint16_t EEPROMPageSize = 0

Definition at line 136 of file stm32303c_eval_eeprom.c.

Referenced by BSP_EEPROM_WriteBuffer(), EEPROM_I2C_Init(), and EEPROM_SPI_Init().

User Manual by  1.7.6.1

STM32303C_EVAL BSP User Manual

Private Variables

STM32303C_EVAL AUDIO

Variables

static AUDIO_DrvTypeDef *	**pAudioDrv** = NULL
I2S_HandleTypeDef	**hAudioOutI2s**

Variable Documentation

I2S_HandleTypeDef hAudioOutI2s

Definition at line **153** of file **stm32303c_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()**, **I2Sx_Init()**, and **I2Sx_MspInit()**.

AUDIO_DrvTypeDef* pAudioDrv = NULL [static]

Definition at line **152** of file **stm32303c_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()**.

STM32303C_EVAL BSP User Manual

Functions

Bus Operation functions

STM32303C-EVAL Common

Functions

static void	**I2Cx_Init** (void) Eval I2Cx Bus initialization.
static void	**I2Cx_WriteData** (uint16_t Addr, uint8_t Reg, uint16_t RegSize, 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, uint16_t RegSize) Read a register of the device through BUS.
static HAL_StatusTypeDef	**I2Cx_ReadBuffer** (uint16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuffer, uint16_t Length) Reads multiple data on the BUS.
static HAL_StatusTypeDef	**I2Cx_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.
static void	**I2Cx_Error** (void) Eval I2Cx error treatment function.
static void	**I2Cx_MspInit** (I2C_HandleTypeDef *hi2c) Initializes I2C MSP.
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.

Function Documentation

static void I2Cx_Error (void) [static]

Eval I2Cx error treatment function.

Return values:
None

Definition at line 728 of file stm32303c_eval.c.

References heval_I2c, and I2Cx_Init().

Referenced by I2Cx_ReadBuffer(), I2Cx_ReadData(), I2Cx_WriteBuffer(), and I2Cx_WriteData().

static void I2Cx_Init (void) [static]

Eval I2Cx Bus initialization.

Return values:
None

Definition at line 598 of file stm32303c_eval.c.

References EVAL_I2Cx, heval_I2c, and I2Cx_MspInit().

Referenced by AUDIO_IO_Init(), EEPROM_I2C_IO_Init(), I2Cx_Error(), and TSENSOR_IO_Init().

static HAL_StatusTypeDef I2Cx_IsDeviceReady (uint16_t DevAddr, uint32_t Trials) [static]

Checks if target device is ready for communication.

Note:
This function is used with Memory devices

Parameters:

DevAddress	Target device address
Trials	Number of trials

Return values:

HAL	status

Definition at line **718** of file **stm32303c_eval.c**.

References **heval_I2c**, and **I2cxTimeout**.

Referenced by **EEPROM_I2C_IO_IsDeviceReady()**, and **TSENSOR_IO_IsDeviceReady()**.

static void I2Cx_MspInit (I2C_HandleTypeDef * hi2c) [static]

Initializes I2C MSP.

Parameters:

hi2c	I2C handle

Return values:

None	

Definition at line **543** of file **stm32303c_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_SCL_SDA_AF,
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 696 of file stm32303c_eval.c.

References heval_I2c, I2Cx_Error(), and I2cxTimeout.

Referenced by EEPROM_I2C_IO_ReadData(), and TSENSOR_IO_Read().

static uint8_t I2Cx_ReadData (uint16_t Addr,
uint8_t Reg,

uint16_t	RegSize
)	[static]

Read a register of the device through BUS.

Parameters:

Addr	Device address on BUS
Reg	The target register address to read
RegSize	The target register size (can be 8BIT or 16BIT)

Return values:

read register value

Definition at line **670** of file **stm32303c_eval.c**.

References **heval_I2c**, **I2Cx_Error()**, and **I2cxTimeout**.

Referenced by **AUDIO_IO_Read()**.

static HAL_StatusTypeDef	**I2Cx_WriteBuffer** (	**uint16_t**	**Addr,**
		uint8_t	**Reg,**
		uint16_t	**RegSize,**
		uint8_t *	**pBuffer,**
		uint16_t	**Length**
	)		[static]

Write a value in a register of the device through BUS.

Parameters:

Addr	Device address on BUS Bus.
Reg	The target register address to write
RegSize	The target register size (can be 8BIT or 16BIT)
pBuffer	The target register value to be written
Length	buffer size to be written

Return values:

None

Definition at line **648** of file **stm32303c_eval.c**.

References **heval_I2c**, **I2Cx_Error()**, and **I2cxTimeout**.

Referenced by **EEPROM_I2C_IO_WriteData()**, and **TSENSOR_IO_Write()**.

static void I2Cx_WriteData (uint16_t Addr,
uint8_t Reg,
uint16_t RegSize,
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
RegSize	The target register size (can be 8BIT or 16BIT)
Value	The target register value to be written

Return values:

None

Definition at line **625** of file **stm32303c_eval.c**.

References **heval_I2c**, **I2Cx_Error()**, and **I2cxTimeout**.

Referenced by **AUDIO_IO_Write()**.

static void SPIx_Error (void) [static]

SPI error treatment function.

Return values:
None

Definition at line **849** of file **stm32303c_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 **780** of file **stm32303c_eval.c**.

References **EVAL_SPIx**, **heval_Spi**, and **SPIx_MspInit()**.

Referenced by **EEPROM_SPI_IO_Init()**, **LCD_IO_Init()**, **SD_IO_Init()**, and **SPIx_Error()**.

static void SPIx_MspInit (SPI_HandleTypeDef * hspi) [static]

Initializes SPI MSP.

Parameters:
hspi SPI handle

Return values:
None

Definition at line **746** of file **stm32303c_eval.c**.

References **EVAL_SPIx_CLK_ENABLE**, **EVAL_SPIx_MISO_MOSI_AF**, **EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE**, **EVAL_SPIx_MISO_MOSI_GPIO_PORT**, **EVAL_SPIx_MISO_PIN**, **EVAL_SPIx_MOSI_PIN**, **EVAL_SPIx_SCK_AF**, **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 **808** of file **stm32303c_eval.c**.

References **heval_Spi**, **SPIx_Error()**, and **SpixTimeout**.

Referenced by **EEPROM_SPI_IO_ReadByte()**, **EEPROM_SPI_IO_ReadData()**, **EEPROM_SPI_IO_WaitEepromStandbyState()**, **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 **831** of file **stm32303c_eval.c**.

References **heval_Spi**, **SPIx_Error()**, and **SpixTimeout**.

Referenced by **EEPROM_SPI_IO_ReadData()**, **EEPROM_SPI_IO_WaitEepromStandbyState()**, **EEPROM_SPI_IO_WriteByte()**, **EEPROM_SPI_IO_WriteData()**, **LCD_IO_ReadData()**, **LCD_IO_WriteMultipleData()**, **LCD_IO_WriteReg()**, and **SD_IO_WriteByte()**.

STM32303C_EVAL BSP User Manual

Private Functions

STM32303C_EVAL AUDIO

Functions

static void	**I2Sx_MspInit** (void) AUDIO OUT I2S MSP Init.
static **AUDIO_StatusTypeDef**	**I2Sx_Init** (uint32_t AudioFreq) Initializes the Audio Codec audio interface (I2S)

Function Documentation

static AUDIO_StatusTypeDef I2Sx_Init (uint32_t AudioFreq) [stati

Initializes the Audio Codec audio interface (I2S)

Parameters:

AudioFreq	Audio frequency to be configured for the I2S peripheral.

Return values:

AUDIO_StatusTypeDef	AUDIO Status

Definition at line **555** of file **stm32303c_eval_audio.c**.

References **AUDIO_ERROR**, **AUDIO_OK**, **hAudioOutI2s**, **I2Sx**, and **I2Sx_MspInit()**.

Referenced by **BSP_AUDIO_OUT_Init()**, and **BSP_AUDIO_OUT_SetFrequency()**.

static void I2Sx_MspInit (void) [static]

AUDIO OUT I2S MSP Init.

Return values:

None

Definition at line **482** of file **stm32303c_eval_audio.c**.

References **AUDIO_OUT_IRQ_PREPRIO**, **AUDIO_OUT_IRQ_SUBPRIO**, **hAudioOutI2s**, **I2Sx_CLK_ENABLE**, **I2Sx_DIN_PIN**, **I2Sx_DMAx_CHANNEL**, **I2Sx_DMAx_CLK_ENABLE**, **I2Sx_DMAx_IRQ**, **I2Sx_DMAx_MEM_DATA_SIZE**, **I2Sx_DMAx_PERIPH_DATA_SIZE**,

I2Sx_FORCE_RESET, **I2Sx_MCK_AF**, **I2Sx_MCK_GPIO_PORT**, **I2Sx_MCK_PIN**, **I2Sx_MCK_WS_GPIO_CLK_ENABLE**, **I2Sx_RELEASE_RESET**, **I2Sx_SCK_DIN_AF**, **I2Sx_SCK_DIN_GPIO_CLK_ENABLE**, **I2Sx_SCK_DIN_GPIO_PORT**, **I2Sx_SCK_PIN**, **I2Sx_WS_AF**, **I2Sx_WS_GPIO_PORT**, and **I2Sx_WS_PIN**.

Referenced by **I2Sx_Init()**.

STM32303C_EVAL BSP User Manual

Private Functions

STM32303C-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 **840** of file **stm32303c_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 **825** of file **stm32303c_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 **910** of file **stm32303c_eval_lcd.c**.

References **lcd_drv**.

Referenced by **BSP_LCD_DrawBitmap()**, and **BSP_LCD_DrawVLine()**.

STM32303C_EVAL BSP User Manual

STM32303C-EVAL LED

Exported Constants

Define for STM32303C_EVAL board. More...

Defines

#define	**LEDn** 4
#define	**LED1_PIN** GPIO_PIN_8
#define	**LED1_GPIO_PORT** GPIOE
#define	**LED1_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_CLK_E
#define	**LED1_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_CLK_D
#define	**LED2_PIN** GPIO_PIN_9
#define	**LED2_GPIO_PORT** GPIOE
#define	**LED2_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_CLK_E
#define	**LED2_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_CLK_D
#define	**LED3_PIN** GPIO_PIN_10
#define	**LED3_GPIO_PORT** GPIOE
#define	**LED3_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_CLK_E
#define	**LED3_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_CLK_D
#define	**LED4_PIN** GPIO_PIN_11
#define	**LED4_GPIO_PORT** GPIOE
#define	**LED4_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_CLK_E
#define	**LED4_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_CLK_D
#define	**LEDx_GPIO_CLK_ENABLE**(__LED__)
#define	**LEDx_GPIO_CLK_DISABLE**(__LED__)

Detailed Description

Define for STM32303C_EVAL board.

Define Documentation

#define LED1_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOE_CLK_

Definition at line **165** of file **stm32303c_eval.h**.

#define LED1_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOE_CLK_

Definition at line **164** of file **stm32303c_eval.h**.

#define LED1_GPIO_PORT GPIOE

Definition at line **163** of file **stm32303c_eval.h**.

#define LED1_PIN GPIO_PIN_8

Definition at line **162** of file **stm32303c_eval.h**.

#define LED2_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOE_CLK_

Definition at line **170** of file **stm32303c_eval.h**.

#define LED2_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOE_CLK_

Definition at line **169** of file **stm32303c_eval.h**.

#define LED2_GPIO_PORT GPIOE

Definition at line **168** of file **stm32303c_eval.h**.

#define LED2_PIN GPIO_PIN_9

Definition at line 167 of file stm32303c_eval.h.

#define LED3_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOE_CLK

Definition at line 175 of file stm32303c_eval.h.

#define LED3_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOE_CLK_

Definition at line 174 of file stm32303c_eval.h.

#define LED3_GPIO_PORT GPIOE

Definition at line 173 of file stm32303c_eval.h.

#define LED3_PIN GPIO_PIN_10

Definition at line 172 of file stm32303c_eval.h.

#define LED4_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOE_CLK

Definition at line 180 of file stm32303c_eval.h.

#define LED4_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOE_CLK_

Definition at line 179 of file stm32303c_eval.h.

#define LED4_GPIO_PORT GPIOE

Definition at line 178 of file stm32303c_eval.h.

#define LED4_PIN GPIO_PIN_11

Definition at line 177 of file stm32303c_eval.h.

#define LEDn 4

Definition at line 160 of file stm32303c_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 187 of file stm32303c_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 **182** of file **stm32303c_eval.h**.

Referenced by **BSP_LED_Init()**.

STM32303C_EVAL BSP User Manual

Private Defines

STM32303C-EVAL LCD

Defines

#define **POLY_X**(Z) ((int32_t)((pPoints + (Z))->X))
#define **POLY_Y**(Z) ((int32_t)((pPoints + (Z))->Y))
#define **MAX_HEIGHT_FONT** 17
#define **MAX_WIDTH_FONT** 24
#define **OFFSET_BITMAP** 54

Define Documentation

#define MAX_HEIGHT_FONT 17

Definition at line 91 of file stm32303c_eval_lcd.c.

#define MAX_WIDTH_FONT 24

Definition at line 92 of file stm32303c_eval_lcd.c.

#define OFFSET_BITMAP 54

Definition at line 93 of file stm32303c_eval_lcd.c.

Referenced by LCD_DrawChar().

#define POLY_X (Z) ((int32_t)((pPoints + (Z))->X))

Definition at line 88 of file stm32303c_eval_lcd.c.

#define POLY_Y (Z) ((int32_t)((pPoints + (Z))->Y))

Definition at line 89 of file stm32303c_eval_lcd.c.

STM32303C_EVAL BSP User Manual

Defines

Exported Constants

STM32303C-EVAL SD

Defines

#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_CMD_CLR_WRITE_PROT 29 /* CMD29 = 0x5D */

Definition at line 225 of file stm32303c_eval_sd.h.

#define SD_CMD_ERASE 38 /* CMD38 = 0x66 */

Definition at line 233 of file stm32303c_eval_sd.h.

Referenced by BSP_SD_Erase().

#define SD_CMD_ERASE_GRP_END 36 /* CMD36 = 0x64 */

Definition at line 231 of file stm32303c_eval_sd.h.

#define SD_CMD_ERASE_GRP_START 35 /* CMD35 = 0x63 */

Definition at line 230 of file stm32303c_eval_sd.h.

#define SD_CMD_GO_IDLE_STATE 0 /* CMD0 = 0x40 */

Commands: CMDxx = CMD-number | 0x40.

Definition at line 211 of file stm32303c_eval_sd.h.

Referenced by SD_GoIdleState().

#define SD_CMD_PROG_CSD 27 /* CMD27 = 0x5B */

Definition at line 223 of file stm32303c_eval_sd.h.

#define SD_CMD_READ_MULT_BLOCK 18 /* CMD18 = 0x52 */

Definition at line 219 of file stm32303c_eval_sd.h.

#define SD_CMD_READ_SINGLE_BLOCK 17 /* CMD17 = 0x51 */

Definition at line 218 of file stm32303c_eval_sd.h.

Referenced by BSP_SD_ReadBlocks().

#define SD_CMD_SD_ERASE_GRP_END 33 /* CMD33 = 0x61 */

Definition at line 228 of file stm32303c_eval_sd.h.

Referenced by BSP_SD_Erase().

#define SD_CMD_SD_ERASE_GRP_START 32 /* CMD32 = 0x60 */

Definition at line 227 of file stm32303c_eval_sd.h.

Referenced by BSP_SD_Erase().

#define SD_CMD_SEND_CID 10 /* CMD10 = 0x4A */

Definition at line 214 of file stm32303c_eval_sd.h.

Referenced by SD_GetCIDRegister().

#define SD_CMD_SEND_CSD 9 /* CMD9 = 0x49 */

Definition at line **213** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

#define SD_CMD_SEND_OP_COND 1 /* CMD1 = 0x41 */

Definition at line **212** of file **stm32303c_eval_sd.h**.

Referenced by **SD_GoIdleState()**.

#define SD_CMD_SEND_STATUS 13 /* CMD13 = 0x4D */

Definition at line **216** of file **stm32303c_eval_sd.h**.

#define SD_CMD_SEND_WRITE_PROT 30 /* CMD30 = 0x5E */

Definition at line **226** of file **stm32303c_eval_sd.h**.

#define SD_CMD_SET_BLOCK_COUNT 23 /* CMD23 = 0x57 */

Definition at line **220** of file **stm32303c_eval_sd.h**.

#define SD_CMD_SET_BLOCKLEN 16 /* CMD16 = 0x50 */

Definition at line **217** of file **stm32303c_eval_sd.h**.

Referenced by **BSP_SD_ReadBlocks()**.

#define SD_CMD_SET_WRITE_PROT 28 /* CMD28 = 0x5C */

Definition at line **224** of file **stm32303c_eval_sd.h**.

#define SD_CMD_STOP_TRANSMISSION 12 /* CMD12 = 0x4C */

Definition at line **215** of file **stm32303c_eval_sd.h**.

#define SD_CMD_UNTAG_ERASE_GROUP 37 /* CMD37 = 0x65 */

Definition at line **232** of file **stm32303c_eval_sd.h**.

#define SD_CMD_UNTAG_SECTOR 34 /* CMD34 = 0x62 */

Definition at line **229** of file **stm32303c_eval_sd.h**.

#define SD_CMD_WRITE_MULT_BLOCK 25 /* CMD25 = 0x59 */

Definition at line **222** of file **stm32303c_eval_sd.h**.

#define SD_CMD_WRITE_SINGLE_BLOCK 24 /* CMD24 = 0x58 */

Definition at line **221** of file **stm32303c_eval_sd.h**.

Referenced by **BSP_SD_WriteBlocks()**.

#define SD_NOT_PRESENT ((uint8_t)0x00)

Definition at line **206** of file **stm32303c_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 205 of file stm32303c_eval_sd.h.

Referenced by BSP_SD_Init(), and BSP_SD_IsDetected().

#define SD_START_DATA_MULTIPLE_BLOCK_READ 0xFE /* Data

Definition at line 197 of file stm32303c_eval_sd.h.

#define SD_START_DATA_MULTIPLE_BLOCK_WRITE 0xFD /* Dat

Definition at line 199 of file stm32303c_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 196 of file stm32303c_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 198 of file stm32303c_eval_sd.h.

Referenced by BSP_SD_WriteBlocks().

#define SD_STOP_DATA_MULTIPLE_BLOCK_WRITE 0xFD /* Data

Definition at line **200** of file **stm32303c_eval_sd.h**.

STM32303C_EVAL BSP User Manual

Private Constants

STM32303C-EVAL SD

Defines

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

Define Documentation

#define SD_DUMMY_BYTE 0xFF

Definition at line **116** of file **stm32303c_eval_sd.c**.

Referenced by **BSP_SD_WriteBlocks()**, **SD_GetCIDRegister()**, and **SD_GetCSDRegister()**.

#define SD_NO_RESPONSE_EXPECTED 0x80

Definition at line **117** of file **stm32303c_eval_sd.c**.

STM32303C_EVAL BSP User Manual

Private Variables

STM32303C-EVAL SD

Variables

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

Variable Documentation

__IO uint8_t SdStatus = SD_PRESENT

Definition at line **129** of file **stm32303c_eval_sd.c**.

Referenced by **BSP_SD_Init()**.

STM32303C_EVAL BSP User Manual

Private Variables

STM32303C-EVAL TSENSOR

Variables

static TSENSOR_DrvTypeDef * **tsensor_drv**

Variable Documentation

TSENSOR_DrvTypeDef* tsensor_drv [static]

Definition at line 78 of file stm32303c_eval_tsensor.c.

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

STM32303C_EVAL BSP User Manual

Enumerations

Exported Types

STM32303C-EVAL TSENSOR

Enumerations

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

Enumeration Type Documentation

enum TSENSOR_Status_TypDef

TSENSOR Status.

Enumerator:

TSENSOR_OK

TSENSOR_ERROR

Definition at line **75** of file **stm32303c_eval_tsensor.h**.

STM32303C_EVAL BSP User Manual

Defines

Exported Constants

STM32303C-EVAL TSENSOR

Defines

#define	**TSENSOR_I2C_ADDRESS** 0x90
#define	**TSENSOR_MAX_TRIALS** 50

Define Documentation

#define TSENSOR_I2C_ADDRESS 0x90

Definition at line 89 of file stm32303c_eval_tsensor.h.

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

#define TSENSOR_MAX_TRIALS 50

Definition at line 92 of file stm32303c_eval_tsensor.h.

Referenced by BSP_TSENSOR_Init().

STM32303C_EVAL BSP User Manual

Firmware Directory Reference

Directories

directory	**Drivers**

STM32303C_EVAL BSP User Manual

Drivers Directory Reference

Directories

directory	**BSP**

STM32303C_EVAL BSP User Manual

BSP Directory Reference

Directories

directory	**STM32303C_EVAL**

STM32303C_EVAL BSP User Manual

STM32303C_EVAL Directory Reference

Files

file **stm32303c_eval.c** [code]

This file provides a set of firmware functions to manage LEDs, push-button and COM port available on STM32303C-EVAL evaluation board (MB1019) RevC from STMicroelectronics.

file **stm32303c_eval.h** [code]

This file contains definitions for STM32303C_EVAL's Leds, push-buttons and COM port hardware resources.

file **stm32303c_eval_audio.c** [code]

This file provides the Audio driver for the STM32303C_EVAL evaluation board(MB1019).

file **stm32303c_eval_audio.h** [code]

This file contains all the functions prototypes for the **stm32303c_eval_audio.c** driver.

file **stm32303c_eval_eeprom.c** [code]

This file provides a set of functions needed to manage a M24LR64 or M24M01 I2C EEPROM memory, or a M95M01 SPI EEPROM memory.

file **stm32303c_eval_eeprom.h** [code]

This file contains all the functions prototypes for the

stm32303c_eval_eeprom.c firmware driver.

file **stm32303c_eval_lcd.c** [code]

This file includes the driver for Liquid Crystal Display modules mounted on STM32303C-EVAL evaluation board.

file **stm32303c_eval_lcd.h** [code]

This file contains all the functions prototypes for the **stm32303c_eval_lcd.c** driver.

file **stm32303c_eval_sd.c** [code]

This file provides a set of functions needed to manage the SPI SD Card memory mounted on STM32303C-EVAL board. It implements a high level communication layer for read and write from/to this memory. The needed STM32F30x hardware resources (SPI and GPIO) are defined in **stm32303c_eval.h** file, and the initialization is performed in SD_LowLevel_Init() function declared in **stm32303c_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and SD_LowLevel_Init() function.

file **stm32303c_eval_sd.h** [code]

This file contains the common defines and functions prototypes for the **stm32303c_eval_sd.c** driver.

file **stm32303c_eval_tsensor.c** [code]

This file provides a set of functions needed to manage the I2C TS751 temperature sensor mounted on STM32303C-EVAL board . It implements a high level communication layer for read and write from/to this sensor. The needed STM323F30x hardware resources (I2C and GPIO) are defined in **stm32303c_eval.h** file, and the initialization is performed in **TSENSOR_IO_Init()** function declared in **stm32303c_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **TSENSOR_IO_Init()** function.

file **stm32303c_eval_tsensor.h** [code]

This file contains all the functions prototypes for the **stm32303c_eval_tsensor.c** firmware driver.

STM32303C_EVAL BSP User Manual

stm32303c_eval.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32303c_eval.h
00004   * @author  MCD Application Team
00005   * @brief   This file contains definitions for STM32303C_EVAL's Leds, push-buttons
00006   *          and COM port hardware resources.
00007   ******************************************************************************
00008   * @attention
00009   *
00010   * <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>
00011   *
00012   * Redistribution and use in source and binary forms, with or without modification,
00013   * are permitted provided that the following conditions are met:
00014   *   1. Redistributions of source code must retain the above copyright notice,
```

```
00015   *      this list of conditions and the fol
lowing disclaimer.
00016   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00017   *      this list of conditions and the fol
lowing disclaimer in the documentation
00018   *      and/or other materials provided wit
h the distribution.
00019   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00020   *      may be used to endorse or promote p
roducts derived from this software
00021   *      without specific prior written perm
ission.
00022   *
00023   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00024   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00025   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00026   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00027   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00028   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
00029   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00030   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00031   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00032   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00033   *
00034   ****************************************
```

```
**********************************
00035   */
00036
00037 /* Define to prevent recursive inclusion ---
-----------------------------------*/
00038 #ifndef __STM32303C_EVAL_H
00039 #define __STM32303C_EVAL_H
00040
00041 #ifdef __cplusplus
00042  extern "C" {
00043 #endif
00044
00045 /** @addtogroup BSP
00046   * @{
00047   */
00048
00049 /** @defgroup STM32303C_EVAL STM32303C-EVAL
00050   * @{
00051   */
00052
00053 /* Includes ------------------------------
-----------------------------------*/
00054 #include "stm32f3xx_hal.h"
00055
00056 /** @defgroup STM32303C_EVAL_Common STM32303
C-EVAL Common
00057   * @{
00058   */
00059
00060 /** @defgroup STM32303C_EVAL_Private_Constan
ts Private Constants
00061   * @{
00062   */
00063 /**
00064 * @}
00065 */
00066
```

```
00067 /** @defgroup STM32303C_EVAL_Private_Variabl
es Private Variables
00068   * @{
00069   */
00070 /**
00071 * @}
00072   */
00073
00074 /** @defgroup STM32303C_EVAL_Exported_Types
Exported Types
00075   * @{
00076   */
00077
00078 /**
00079  * @brief LED Types Definition
00080  */
00081 typedef enum
00082 {
00083   LED1 = 0,
00084   LED2 = 1,
00085   LED3 = 2,
00086   LED4 = 3,
00087   /* Color led aliases */
00088   LED_GREEN  = LED1,
00089   LED_ORANGE = LED2,
00090   LED_RED    = LED3,
00091   LED_BLUE   = LED4
00092 }Led_TypeDef;
00093
00094 /**
00095  * @brief BUTTON Types Definition
00096  */
00097 typedef enum
00098 {
00099   BUTTON_KEY   = 0,
00100   BUTTON_SEL   = 1,
00101   BUTTON_LEFT  = 2,
```

```
00102   BUTTON_RIGHT = 3,
00103   BUTTON_DOWN  = 4,
00104   BUTTON_UP    = 5
00105 }Button_TypeDef;
00106
00107 typedef enum
00108 {
00109   BUTTON_MODE_GPIO = 0,
00110   BUTTON_MODE_EXTI = 1
00111 }ButtonMode_TypeDef;
00112
00113 /**
00114  * @brief JOYSTICK Types Definition
00115  */
00116 typedef enum
00117 {
00118   JOY_SEL   = 0,
00119   JOY_LEFT  = 1,
00120   JOY_RIGHT = 2,
00121   JOY_DOWN  = 3,
00122   JOY_UP    = 4,
00123   JOY_NONE  = 5
00124 }JOYState_TypeDef;
00125
00126 typedef enum
00127 {
00128   JOY_MODE_GPIO = 0,
00129   JOY_MODE_EXTI = 1
00130 }JOYMode_TypeDef;
00131
00132 /**
00133  * @brief COM Types Definition
00134  */
00135 typedef enum
00136 {
00137   COM1 = 0
00138 }COM_TypeDef;
```

```
00139
00140 /**
00141   * @}
00142   */
00143
00144 /** @defgroup STM32303C_EVAL_Exported_Consta
nts Exported Constants
00145   * @{
00146   */
00147
00148 /**
00149   * @brief  Define for STM32303C_EVAL board

00150   */
00151
00152 #if !defined (USE_STM32303C_EVAL)
00153  #define USE_STM32303C_EVAL
00154 #endif
00155
00156
00157 /** @defgroup STM32303C_EVAL_LED STM32303C-E
VAL LED
00158   * @{
00159   */
00160 #define LEDn                                4
00161
00162 #define LED1_PIN                            GPI
O_PIN_8
00163 #define LED1_GPIO_PORT                      GPI
OE
00164 #define LED1_GPIO_CLK_ENABLE()              __H
AL_RCC_GPIOE_CLK_ENABLE()
00165 #define LED1_GPIO_CLK_DISABLE()             __H
AL_RCC_GPIOE_CLK_DISABLE()
00166
00167 #define LED2_PIN                            GPI
O_PIN_9
```

```
00168 #define LED2_GPIO_PORT                          GPI
OE
00169 #define LED2_GPIO_CLK_ENABLE()                  __H
AL_RCC_GPIOE_CLK_ENABLE()
00170 #define LED2_GPIO_CLK_DISABLE()                 __H
AL_RCC_GPIOE_CLK_DISABLE()
00171
00172 #define LED3_PIN                                GPI
O_PIN_10
00173 #define LED3_GPIO_PORT                          GPI
OE
00174 #define LED3_GPIO_CLK_ENABLE()                  __H
AL_RCC_GPIOE_CLK_ENABLE()
00175 #define LED3_GPIO_CLK_DISABLE()                 __H
AL_RCC_GPIOE_CLK_DISABLE()
00176
00177 #define LED4_PIN                                GPI
O_PIN_11
00178 #define LED4_GPIO_PORT                          GPI
OE
00179 #define LED4_GPIO_CLK_ENABLE()                  __H
AL_RCC_GPIOE_CLK_ENABLE()
00180 #define LED4_GPIO_CLK_DISABLE()                 __H
AL_RCC_GPIOE_CLK_DISABLE()
00181
00182 #define LEDx_GPIO_CLK_ENABLE(__LED__)    do
{ if ((__LED__) == LED1) LED1_GPIO_CLK_ENABLE(); e
lse\
00183
  if ((__LED__) == LED2) LED2_GPIO_CLK_ENABLE(); e
lse\
00184
  if ((__LED__) == LED3) LED3_GPIO_CLK_ENABLE(); e
lse\
00185
  if ((__LED__) == LED4) LED4_GPIO_CLK_ENABLE();}
while(0)
```

```
00186
00187 #define LEDx_GPIO_CLK_DISABLE(__LED__)   (((
__LED__) == LED1) ? LED1_GPIO_CLK_DISABLE() :\
00188                                             ((
__LED__) == LED2) ? LED2_GPIO_CLK_DISABLE() :\
00189                                             ((
__LED__) == LED3) ? LED3_GPIO_CLK_DISABLE() :\
00190                                             ((
__LED__) == LED4) ? LED4_GPIO_CLK_DISABLE() : 0 )
00191
00192 /**
00193   * @}
00194   */
00195
00196 /** @defgroup STM32303C_EVAL_BUTTON STM32303
C-EVAL BUTTON
00197   * @{
00198   */
00199 #define JOYn                                5
00200 #define BUTTONn                             1 +
 JOYn
00201
00202 /**
00203  * @brief Key push-button
00204  */
00205 #define KEY_BUTTON_PIN                      GPI
O_PIN_6
00206 #define KEY_BUTTON_GPIO_PORT                GPI
OE
00207 #define KEY_BUTTON_GPIO_CLK_ENABLE()        __H
AL_RCC_GPIOE_CLK_ENABLE()
00208 #define KEY_BUTTON_GPIO_CLK_DISABLE()       __H
AL_RCC_GPIOE_CLK_DISABLE()
00209 #define KEY_BUTTON_EXTI_IRQn                EXT
I9_5_IRQn
00210
00211 /**
```

```
00212  * @brief Joystick Right push-button
00213  */
00214 #define RIGHT_JOY_PIN                       GPI
O_PIN_6
00215 #define RIGHT_JOY_GPIO_PORT                 GPI
OD
00216 #define RIGHT_JOY_GPIO_CLK_ENABLE()         __H
AL_RCC_GPIOD_CLK_ENABLE()
00217 #define RIGHT_JOY_GPIO_CLK_DISABLE()        __H
AL_RCC_GPIOD_CLK_DISABLE()
00218 #define RIGHT_JOY_EXTI_IRQn                 EXT
I9_5_IRQn
00219
00220 /**
00221  * @brief Joystick Left push-button
00222  */
00223 #define LEFT_JOY_PIN                        GPI
O_PIN_5
00224 #define LEFT_JOY_GPIO_PORT                  GPI
OB
00225 #define LEFT_JOY_GPIO_CLK_ENABLE()          __H
AL_RCC_GPIOB_CLK_ENABLE()
00226 #define LEFT_JOY_GPIO_CLK_DISABLE()         __H
AL_RCC_GPIOB_CLK_DISABLE()
00227 #define LEFT_JOY_EXTI_IRQn                  EXT
I9_5_IRQn
00228
00229 /**
00230  * @brief Joystick Up push-button
00231  */
00232 #define UP_JOY_PIN                          GPI
O_PIN_7
00233 #define UP_JOY_GPIO_PORT                    GPI
OE
00234 #define UP_JOY_GPIO_CLK_ENABLE()            __H
AL_RCC_GPIOE_CLK_ENABLE()
00235 #define UP_JOY_GPIO_CLK_DISABLE()           __H
```

```
AL_RCC_GPIOE_CLK_DISABLE()
00236 #define UP_JOY_EXTI_IRQn                 EXT
I9_5_IRQn
00237
00238 /**
00239  * @brief Joystick Down push-button
00240  */
00241 #define DOWN_JOY_PIN                     GPI
O_PIN_5
00242 #define DOWN_JOY_GPIO_PORT               GPI
OD
00243 #define DOWN_JOY_GPIO_CLK_ENABLE()       __H
AL_RCC_GPIOD_CLK_ENABLE()
00244 #define DOWN_JOY_GPIO_CLK_DISABLE()      __H
AL_RCC_GPIOD_CLK_DISABLE()
00245 #define DOWN_JOY_EXTI_IRQn               EXT
I9_5_IRQn
00246
00247 /**
00248  * @brief Joystick Sel push-button
00249  */
00250 #define SEL_JOY_PIN                      GPI
O_PIN_13
00251 #define SEL_JOY_GPIO_PORT                GPI
OC
00252 #define SEL_JOY_GPIO_CLK_ENABLE()        __H
AL_RCC_GPIOC_CLK_ENABLE()
00253 #define SEL_JOY_GPIO_CLK_DISABLE()       __H
AL_RCC_GPIOC_CLK_DISABLE()
00254 #define SEL_JOY_EXTI_IRQn                EXT
I15_10_IRQn
00255
00256 #define BUTTONx_GPIO_CLK_ENABLE(__BUTTON__)
    do { if ((__BUTTON__) == BUTTON_KEY) KEY_BUTTO
N_GPIO_CLK_ENABLE(); else\
00257
         if ((__BUTTON__) == BUTTON_SEL) SEL_JOY_G
```

```
PIO_CLK_ENABLE(); else\
00258
         if ((__BUTTON__) == BUTTON_LEFT) LEFT_JOY
_GPIO_CLK_ENABLE(); else\
00259
         if ((__BUTTON__) == BUTTON_RIGHT) RIGHT_J
OY_GPIO_CLK_ENABLE(); else\
00260
         if ((__BUTTON__) == BUTTON_DOWN) DOWN_JOY
_GPIO_CLK_ENABLE(); else\
00261
         if ((__BUTTON__) == BUTTON_UP) UP_JOY_GPI
O_CLK_ENABLE();} while(0)
00262
00263 #define BUTTONx_GPIO_CLK_DISABLE(__BUTTON__)
    (((__BUTTON__) == BUTTON_KEY) ? KEY_BUTTON_GPI
O_CLK_DISABLE()  :\
00264
     ((__BUTTON__) == BUTTON_SEL) ? SEL_JOY_GPIO_C
LK_DISABLE()  :\
00265
     ((__BUTTON__) == BUTTON_LEFT) ? LEFT_JOY_GPIO
_CLK_DISABLE()  :\
00266
     ((__BUTTON__) == BUTTON_RIGHT) ? RIGHT_JOY_GP
IO_CLK_DISABLE()  :\
00267
     ((__BUTTON__) == BUTTON_DOWN) ? DOWN_JOY_GPIO
_CLK_DISABLE()  :\
00268
     ((__BUTTON__) == BUTTON_UP) ? UP_JOY_GPIO_CLK
_DISABLE() : 0 )
00269
00270 #define JOYx_GPIO_CLK_ENABLE(__JOY__)
   do { if ((__JOY__) == JOY_SEL) SEL_JOY_GPIO_CLK
_ENABLE(); else\
00271
```

```
        if ((__JOY__) == JOY_LEFT) LEFT_JOY_GPIO_C
LK_ENABLE(); else\
00272
        if ((__JOY__) == JOY_RIGHT) RIGHT_JOY_GPIO
_CLK_ENABLE(); else\
00273
        if ((__JOY__) == JOY_DOWN) DOWN_JOY_GPIO_C
LK_ENABLE(); else\
00274
        if ((__JOY__) == JOY_UP) UP_JOY_GPIO_CLK_E
NABLE();} while(0)
00275
00276 #define JOYx_GPIO_CLK_DISABLE(__JOY__)
   (((__JOY__) == JOY_SEL) ? SEL_JOY_GPIO_CLK_DISA
BLE() :\
00277
    ((__JOY__) == JOY_LEFT) ? LEFT_JOY_GPIO_CLK_DI
SABLE() :\
00278
    ((__JOY__) == JOY_RIGHT) ? RIGHT_JOY_GPIO_CLK_
DISABLE() :\
00279
    ((__JOY__) == JOY_DOWN) ? DOWN_JOY_GPIO_CLK_DI
SABLE() :\
00280
    ((__JOY__) == JOY_UP) ? UP_JOY_GPIO_CLK_DISABL
E() : 0 )
00281
00282 /**
00283   * @}
00284   */
00285
00286 /** @defgroup STM32303C_EVAL_COM STM32303C-E
VAL COM
00287   * @{
00288   */
00289 #if defined(HAL_UART_MODULE_ENABLED)
```

```
00290
00291 #define COMn
    1
00292
00293 /**
00294  * @brief Definition for COM port1, connecte
d to USART1
00295  */
00296 #define EVAL_COM1
    USART1
00297 #define EVAL_COM1_CLK_ENABLE()
    __HAL_RCC_USART1_CLK_ENABLE()
00298 #define EVAL_COM1_CLK_DISABLE()
    __HAL_RCC_USART1_CLK_DISABLE()
00299
00300 #define EVAL_COM1_TX_PIN
    GPIO_PIN_4
00301 #define EVAL_COM1_TX_GPIO_PORT
    GPIOC
00302 #define EVAL_COM1_TX_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOC_CLK_ENABLE()
00303 #define EVAL_COM1_TX_GPIO_CLK_DISABLE()
    __HAL_RCC_GPIOC_CLK_DISABLE()
00304 #define EVAL_COM1_TX_AF
    GPIO_AF7_USART1
00305
00306 #define EVAL_COM1_RX_PIN
    GPIO_PIN_1
00307 #define EVAL_COM1_RX_GPIO_PORT
    GPIOE
00308 #define EVAL_COM1_RX_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOE_CLK_ENABLE()
00309 #define EVAL_COM1_RX_GPIO_CLK_DISABLE()
    __HAL_RCC_GPIOE_CLK_DISABLE()
00310 #define EVAL_COM1_RX_AF
    GPIO_AF7_USART1
00311
```

```
00312 #define EVAL_COM1_IRQn
    USART1_IRQn
00313
00314 #define COMx_CLK_ENABLE(__INDEX__)
    do { if ((__INDEX__) == COM1) EVAL_COM1_CLK_EN
ABLE();} while(0)
00315 #define COMx_CLK_DISABLE(__INDEX__)
    (((__INDEX__) == COM1) ? EVAL_COM1_CLK_DISABLE
() : 0)
00316
00317 #define COMx_TX_GPIO_CLK_ENABLE(__INDEX__)
    do { if ((__INDEX__) == COM1) EVAL_COM1_TX_GPI
O_CLK_ENABLE();} while(0)
00318 #define COMx_TX_GPIO_CLK_DISABLE(__INDEX__)
    (((__INDEX__) == COM1) ? EVAL_COM1_TX_GPIO_CLK
_DISABLE() : 0)
00319
00320 #define COMx_RX_GPIO_CLK_ENABLE(__INDEX__)
    do { if ((__INDEX__) == COM1) EVAL_COM1_RX_GPI
O_CLK_ENABLE();} while(0)
00321 #define COMx_RX_GPIO_CLK_DISABLE(__INDEX__)
    (((__INDEX__) == COM1) ? EVAL_COM1_RX_GPIO_CLK
_DISABLE() : 0)
00322
00323 #endif /* HAL_UART_MODULE_ENABLED */
00324
00325 #if defined(HAL_I2C_MODULE_ENABLED)
00326 /*####################### I2Cx ###############
#####################*/
00327 /* User can use this section to tailor I2Cx
instance used and associated
00328    resources */
00329 /* Definition for I2Cx Pins */
00330 #define EVAL_I2Cx_SCL_PIN
    GPIO_PIN_6
00331 #define EVAL_I2Cx_SCL_GPIO_PORT
    GPIOF
```

```
00332 #define EVAL_I2Cx_SDA_PIN
    GPIO_PIN_10
00333 #define EVAL_I2Cx_SDA_GPIO_PORT
    GPIOA
00334 #define EVAL_I2Cx_SCL_SDA_AF
    GPIO_AF4_I2C2
00335
00336 /* Definition for I2Cx clock resources */
00337 #define EVAL_I2Cx
    I2C2
00338 #define EVAL_I2Cx_CLK_ENABLE()
    __HAL_RCC_I2C2_CLK_ENABLE()
00339 #define EVAL_I2Cx_SDA_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOA_CLK_ENABLE()
00340 #define EVAL_I2Cx_SCL_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOF_CLK_ENABLE()
00341
00342 #define EVAL_I2Cx_FORCE_RESET()
    __HAL_RCC_I2C2_FORCE_RESET()
00343 #define EVAL_I2Cx_RELEASE_RESET()
    __HAL_RCC_I2C2_RELEASE_RESET()
00344
00345 /* Definition for I2Cx's NVIC */
00346 #define EVAL_I2Cx_EV_IRQn
    I2C2_EV_IRQn
00347 #define EVAL_I2Cx_EV_IRQHandler
    I2C2_EV_IRQHandler
00348 #define EVAL_I2Cx_ER_IRQn
    I2C2_ER_IRQn
00349 #define EVAL_I2Cx_ER_IRQHandler
    I2C2_ER_IRQHandler
00350
00351 /* I2C TIMING Register define when I2C clock
 source is SYSCLK */
00352 /* I2C TIMING is calculated in case of the I
2C Clock source is the SYSCLK = 72 MHz */
00353 /* Set 0xC062121F value to reach 100 KHz spe
```

```
ed (Rise time = 640ns, Fall time = 20ns) */
00354 #ifndef EVAL_I2Cx_TIMING
00355  #define EVAL_I2Cx_TIMING
    0xC062121F
00356 #endif /* EVAL_I2Cx_TIMING */
00357
00358 /* Maximum Timeout values for flags waiting
loops. These timeouts are not based
00359    on accurate values, they just guarantee t
hat the application will not remain
00360    stuck if the I2C communication is corrupt
ed.
00361    You may modify these timeout values depen
ding on CPU frequency and application
00362    conditions (interrupts routines ...). */

00363 #define EVAL_I2Cx_TIMEOUT_MAX
    1000
00364
00365 #endif /* HAL_I2C_MODULE_ENABLED */
00366
00367 #if defined(HAL_SPI_MODULE_ENABLED)
00368 /**
00369   * @brief  Definition for SPI Interface pin
s (SPI2 used)
00370   */
00371 #define EVAL_SPIx
    SPI2
00372 #define EVAL_SPIx_CLK_ENABLE()
    __HAL_RCC_SPI2_CLK_ENABLE()
00373 #define EVAL_SPIx_CLK_DISABLE()
    __HAL_RCC_SPI2_CLK_DISABLE()
00374 #define EVAL_SPIx_FORCE_RESET()
    __HAL_RCC_SPI2_FORCE_RESET()
00375 #define EVAL_SPIx_RELEASE_RESET()
    __HAL_RCC_SPI2_RELEASE_RESET()
00376
```

```
00377 #define EVAL_SPIx_SCK_AF
    GPIO_AF5_SPI2
00378 #define EVAL_SPIx_SCK_GPIO_PORT
    GPIOF
00379 #define EVAL_SPIx_SCK_PIN
    GPIO_PIN_9
00380 #define EVAL_SPIx_SCK_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOF_CLK_ENABLE()
00381 #define EVAL_SPIx_SCK_GPIO_CLK_DISABLE()
    __HAL_RCC_GPIOF_CLK_DISABLE()
00382
00383 #define EVAL_SPIx_MISO_MOSI_AF
    GPIO_AF5_SPI2
00384 #define EVAL_SPIx_MISO_MOSI_GPIO_PORT
    GPIOB
00385 #define EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE(
)   __HAL_RCC_GPIOB_CLK_ENABLE()
00386 #define EVAL_SPIx_MISO_MOSI_GPIO_CLK_DISABLE
()  __HAL_RCC_GPIOB_CLK_DISABLE()
00387 #define EVAL_SPIx_MISO_PIN
    GPIO_PIN_14
00388 #define EVAL_SPIx_MOSI_PIN
    GPIO_PIN_15
00389
00390 /* Maximum Timeout values for flags waiting
loops. These timeouts are not based
00391    on accurate values, they just guarantee t
hat the application will not remain
00392    stuck if the SPI communication is corrupt
ed.
00393    You may modify these timeout values depen
ding on CPU frequency and application
00394    conditions (interrupts routines ...). */

00395 #define EVAL_SPIx_TIMEOUT_MAX
    1000
00396
```

```
00397 #endif /* HAL_SPI_MODULE_ENABLED */
00398 /**
00399   * @}
00400   */
00401
00402
00403 /** @defgroup STM32303C_EVAL_COMPONENT STM32
303C-EVAL COMPONENT
00404   * @{
00405   */
00406 /*####################### LCD ###############
###################*/
00407 /* Chip Select macro definition */
00408 #define LCD_CS_LOW()                          HA
L_GPIO_WritePin(LCD_NCS_GPIO_PORT, LCD_NCS_PIN, GP
IO_PIN_RESET)
00409 #define LCD_CS_HIGH()                         HA
L_GPIO_WritePin(LCD_NCS_GPIO_PORT, LCD_NCS_PIN, GP
IO_PIN_SET)
00410
00411 /**
00412   * @brief  LCD Control Interface pins
00413   */
00414 #define LCD_NCS_PIN                           GP
IO_PIN_0                  /* PE.00 */
00415 #define LCD_NCS_GPIO_PORT                     GP
IOE
00416 #define LCD_NCS_GPIO_CLK_ENABLE()             __
HAL_RCC_GPIOE_CLK_ENABLE()
00417 #define LCD_NCS_GPIO_CLK_DISABLE()            __
HAL_RCC_GPIOE_CLK_DISABLE()
00418
00419 /*####################### SD ################
###################*/
00420 /* Chip Select macro definition */
00421 #define SD_CS_LOW()                           HA
L_GPIO_WritePin(SD_CS_GPIO_PORT, SD_CS_PIN, GPIO_P
```

```
IN_RESET)
00422 #define SD_CS_HIGH()                        HA
L_GPIO_WritePin(SD_CS_GPIO_PORT, SD_CS_PIN, GPIO_P
IN_SET)
00423
00424 /**
00425   * @brief SD card Control pins
00426   */
00427 #define SD_CS_PIN                           GP
IO_PIN_15                  /* PE.15 */
00428 #define SD_CS_GPIO_PORT                     GP
IOE
00429 #define SD_CS_GPIO_CLK_ENABLE()             __
HAL_RCC_GPIOE_CLK_ENABLE()
00430 #define SD_CS_GPIO_CLK_DISABLE()            __
HAL_RCC_GPIOE_CLK_DISABLE()
00431
00432 /**
00433   * @brief  SD Detect Interface pins
00434   */
00435 #define SD_DETECT_PIN                       GP
IO_PIN_6                  /* PC.06 */
00436 #define SD_DETECT_GPIO_PORT                 GP
IOC
00437 #define SD_DETECT_GPIO_CLK_ENABLE()         __
HAL_RCC_GPIOC_CLK_ENABLE()
00438 #define SD_DETECT_GPIO_CLK_DISABLE()        __
HAL_RCC_GPIOC_CLK_DISABLE()
00439 #define SD_DETECT_EXTI_IRQn                 EX
TI9_5_IRQn
00440
00441 /*###################### EEPROM SPI #########
############################*/
00442 /* Chip Select macro definition */
00443 #define EEPROM_CS_LOW()       HAL_GPIO_Write
Pin(EEPROM_CS_GPIO_PORT, EEPROM_CS_PIN, GPIO_PIN_R
ESET)
```

```
00444 #define EEPROM_CS_HIGH()      HAL_GPIO_Write
Pin(EEPROM_CS_GPIO_PORT, EEPROM_CS_PIN, GPIO_PIN_S
ET)
00445
00446 /**
00447   * @brief  EEPROM Control Interface pins
00448   */
00449 #define EEPROM_CS_PIN
    GPIO_PIN_7
00450 #define EEPROM_CS_GPIO_PORT
    GPIOD
00451 #define EEPROM_CS_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOD_CLK_ENABLE()
00452 #define EEPROM_CS_GPIO_CLK_DISABLE()
    __HAL_RCC_GPIOD_CLK_DISABLE()
00453 /**
00454   * @}
00455   */
00456
00457 /**
00458   * @}
00459   */
00460
00461 /** @defgroup STM32303C_EVAL_Exported_Functi
ons Exported Functions
00462   * @{
00463   */
00464 uint32_t          BSP_GetVersion(void);
00465 void              BSP_LED_Init(Led_TypeDef L
ed);
00466 void              BSP_LED_On(Led_TypeDef Led
);
00467 void              BSP_LED_Off(Led_TypeDef Le
d);
00468 void              BSP_LED_Toggle(Led_TypeDef
 Led);
00469 void              BSP_PB_Init(Button_TypeDef
```

```
 Button, ButtonMode_TypeDef Button_Mode);
00470 uint32_t           BSP_PB_GetState(Button_Typ
eDef Button);
00471 uint8_t           BSP_JOY_Init(JOYMode_TypeD
ef Joy_Mode);
00472 JOYState_TypeDef  BSP_JOY_GetState(void);
00473 #if defined(HAL_UART_MODULE_ENABLED)
00474 void              BSP_COM_Init(COM_TypeDef C
OM, UART_HandleTypeDef* huart);
00475 #endif /* HAL_UART_MODULE_ENABLED */
00476
00477 /**
00478   * @}
00479   */
00480
00481 /**
00482   * @}
00483   */
00484
00485 /**
00486   * @}
00487   */
00488
00489 /**
00490   * @}
00491   */
00492
00493 #ifdef __cplusplus
00494 }
00495 #endif
00496
00497 #endif /* __H */
00498
00499 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

Generated on Wed May 31 2017 10:00:44 for STM32303C_EVAL BSP
User Manual by  1.7.6.1

STM32303C_EVAL BSP User Manual

stm32303c_eval.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32303c_eval.c
00004   * @author  MCD Application Team
00005   * @brief   This file provides a set of firmware functions to manage LEDs,
00006   *          push-button and COM port available on STM32303C-EVAL evaluation
00007   *          board (MB1019) RevC from STMicroelectronics.
00008   ******************************************************************************
00009   * @attention
00010   *
00011   * <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>
00012   *
00013   * Redistribution and use in source and binary forms, with or without modification,
00014   * are permitted provided that the following conditions are met:
```

```
00015   *   1. Redistributions of source code must
 retain the above copyright notice,
00016   *      this list of conditions and the fol
lowing disclaimer.
00017   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00018   *      this list of conditions and the fol
lowing disclaimer in the documentation
00019   *      and/or other materials provided wit
h the distribution.
00020   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00021   *      may be used to endorse or promote p
roducts derived from this software
00022   *      without specific prior written perm
ission.
00023   *
00024   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00025   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00026   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00027   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00028   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00029   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
00030   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00031   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00032   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00033   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
```

```
00034   *
00035   ******************************************
*********************************
00036   */
00037
00038 /* Includes ------------------------------
-----------------------------------*/
00039 #include "stm32303c_eval.h"
00040
00041 /** @addtogroup BSP
00042   * @{
00043   */
00044
00045 /** @addtogroup STM32303C_EVAL
00046  * @brief This file provides firmware functi
ons to manage Leds, push-buttons,
00047  *        COM ports, SD card and temperature
 sensor available on
00048  *        STM32303C-EVAL evaluation board fr
om STMicroelectronics.
00049   * @{
00050   */
00051
00052 /** @addtogroup STM32303C_EVAL_Common
00053   * @{
00054   */
00055
00056 /** @addtogroup STM32303C_EVAL_Private_Const
ants
00057   * @{
00058   */
00059 /* LINK LCD */
00060 #define START_BYTE          0x70
00061 #define SET_INDEX           0x00
00062 #define READ_STATUS         0x01
00063 #define LCD_WRITE_REG       0x02
00064 #define LCD_READ_REG        0x03
```

```
00065
00066 /* LINK SD Card */
00067 #define SD_DUMMY_BYTE            0xFF
00068 #define SD_NO_RESPONSE_EXPECTED  0x80
00069
00070 /* LINK EEPROM SPI */
00071 #define EEPROM_CMD_WREN          0x06  /*!< W
rite enable instruction */
00072 #define EEPROM_CMD_WRDI          0x04  /*!< W
rite disable instruction */
00073 #define EEPROM_CMD_RDSR          0x05  /*!< R
ead Status Register instruction  */
00074 #define EEPROM_CMD_WRSR          0x01  /*!< W
rite Status Register instruction */
00075 #define EEPROM_CMD_WRITE         0x02  /*!< W
rite to Memory instruction */
00076 #define EEPROM_CMD_READ          0x03  /*!< R
ead from Memory instruction */
00077
00078 #define EEPROM_WIP_FLAG          0x01  /*!< W
rite In Progress (WIP) flag */
00079
00080 /**
00081  * @brief STM32303C EVAL BSP Driver version
number V2.1.2
00082    */
00083 #define __STM32303C_EVAL_BSP_VERSION_MAIN
    (0x02) /*!< [31:24] main version */
00084 #define __STM32303C_EVAL_BSP_VERSION_SUB1
    (0x01) /*!< [23:16] sub1 version */
00085 #define __STM32303C_EVAL_BSP_VERSION_SUB2
    (0x02) /*!< [15:8]  sub2 version */
00086 #define __STM32303C_EVAL_BSP_VERSION_RC
    (0x00) /*!< [7:0]  release candidate */
00087 #define __STM32303C_EVAL_BSP_VERSION
    ((__STM32303C_EVAL_BSP_VERSION_MAIN << 24)\
00088
```

```
    |(__STM32303C_EVAL_BSP_VERSION_SUB1 << 16)\
00089
    |(__STM32303C_EVAL_BSP_VERSION_SUB2 << 8 )\
00090
    |(__STM32303C_EVAL_BSP_VERSION_RC))
00091 /**
00092   * @}
00093   */
00094
00095 /** @addtogroup STM32303C_EVAL_Private_Varia
bles
00096   * @{
00097   */
00098 /**
00099  * @brief LED variables
00100  */
00101 GPIO_TypeDef* LED_PORT[LEDn] = {LED1_GPIO_PO
RT,
00102                                 LED2_GPIO_PO
RT,
00103                                 LED3_GPIO_PO
RT,
00104                                 LED4_GPIO_PO
RT};
00105
00106 const uint16_t LED_PIN[LEDn] = {LED1_PIN,
00107                                 LED2_PIN,
00108                                 LED3_PIN,
00109                                 LED4_PIN};
00110
00111 /**
00112  * @brief BUTTON variables
00113  */
00114 GPIO_TypeDef* BUTTON_PORT[BUTTONn] = {KEY_BU
TTON_GPIO_PORT,
00115                                       SEL_JO
Y_GPIO_PORT,
```

```
00116                                                   LEFT_J
OY_GPIO_PORT,
00117                                                   RIGHT_
JOY_GPIO_PORT,
00118                                                   DOWN_J
OY_GPIO_PORT,
00119                                                   UP_JOY
_GPIO_PORT};
00120
00121 const uint16_t BUTTON_PIN[BUTTONn] = {KEY_BU
TTON_PIN,
00122                                                   SEL_JO
Y_PIN,
00123                                                   LEFT_J
OY_PIN,
00124                                                   RIGHT_
JOY_PIN,
00125                                                   DOWN_J
OY_PIN,
00126                                                   UP_JOY
_PIN};
00127
00128 const uint16_t BUTTON_IRQn[BUTTONn] = {KEY_B
UTTON_EXTI_IRQn,
00129                                                    SEL_J
OY_EXTI_IRQn,
00130                                                    LEFT_
JOY_EXTI_IRQn,
00131                                                    RIGHT
_JOY_EXTI_IRQn,
00132                                                    DOWN_
JOY_EXTI_IRQn,
00133                                                    UP_JO
Y_EXTI_IRQn};
00134
00135 /**
00136  * @brief JOYSTICK variables
```

```
00137  */
00138 GPIO_TypeDef* JOY_PORT[JOYn] = {SEL_JOY_GPIO
_PORT,
00139                                 LEFT_JOY_GPI
O_PORT,
00140                                 RIGHT_JOY_GP
IO_PORT,
00141                                 DOWN_JOY_GPI
O_PORT,
00142                                 UP_JOY_GPIO_
PORT};
00143
00144 const uint16_t JOY_PIN[JOYn] = {SEL_JOY_PIN,

00145                                 LEFT_JOY_PIN
,
00146                                 RIGHT_JOY_PIN
,
00147                                 DOWN_JOY_PIN
,
00148                                 UP_JOY_PIN};

00149
00150 const uint8_t JOY_IRQn[JOYn] = {SEL_JOY_EXTI
_IRQn,
00151                                 LEFT_JOY_EXT
I_IRQn,
00152                                 RIGHT_JOY_EX
TI_IRQn,
00153                                 DOWN_JOY_EXT
I_IRQn,
00154                                 UP_JOY_EXTI_
IRQn};
00155
00156 /**
00157  * @brief COM variables
00158  */
```

```
00159 #if defined(HAL_UART_MODULE_ENABLED)
00160 USART_TypeDef* COM_USART[COMn]   = {EVAL_COM1
};
00161
00162 GPIO_TypeDef*  COM_TX_PORT[COMn] = {EVAL_COM
1_TX_GPIO_PORT};
00163
00164 GPIO_TypeDef*  COM_RX_PORT[COMn] = {EVAL_COM
1_RX_GPIO_PORT};
00165
00166 const uint16_t COM_TX_PIN[COMn]  = {EVAL_COM
1_TX_PIN};
00167
00168 const uint16_t COM_RX_PIN[COMn]  = {EVAL_COM
1_RX_PIN};
00169
00170 const uint16_t COM_TX_AF[COMn]   = {EVAL_COM
1_TX_AF};
00171
00172 const uint16_t COM_RX_AF[COMn]   = {EVAL_COM
1_RX_AF};
00173 #endif /* HAL_UART_MODULE_ENABLED) */
00174
00175 /**
00176  * @brief BUS variables
00177  */
00178 #if defined(HAL_I2C_MODULE_ENABLED)
00179 uint32_t I2cxTimeout = EVAL_I2Cx_TIMEOUT_MAX
;   /*<! Value of Timeout when I2C communication f
ails */
00180 I2C_HandleTypeDef heval_I2c;
00181 #endif /* HAL_I2C_MODULE_ENABLED */
00182
00183 #if defined(HAL_SPI_MODULE_ENABLED)
00184 uint32_t SpixTimeout = EVAL_SPIx_TIMEOUT_MAX
;        /*<! Value of Timeout when SPI communicat
ion fails */
```

```
00185 static SPI_HandleTypeDef heval_Spi;
00186 #endif /* HAL_SPI_MODULE_ENABLED */
00187
00188 /**
00189   * @}
00190   */
00191
00192 /** @defgroup STM32303C_EVAL_BUS Bus Operati
on functions
00193   * @{
00194   */
00195 #if defined(HAL_I2C_MODULE_ENABLED)
00196 /* I2Cx bus function */
00197 static void               I2Cx_Init(void);
00198 static void               I2Cx_WriteData(uin
t16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t
 Value);
00199 static HAL_StatusTypeDef  I2Cx_WriteBuffer(u
int16_t Addr, uint8_t Reg, uint16_t RegSize, uint8
_t *pBuffer, uint16_t Length);
00200 static uint8_t            I2Cx_ReadData(uint
16_t Addr, uint8_t Reg, uint16_t RegSize);
00201 static HAL_StatusTypeDef  I2Cx_ReadBuffer(ui
nt16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_
t *pBuffer, uint16_t Length);
00202 static HAL_StatusTypeDef  I2Cx_IsDeviceReady
(uint16_t DevAddress, uint32_t Trials);
00203 static void               I2Cx_Error (void);
00204 static void               I2Cx_MspInit(I2C_H
andleTypeDef *hi2c);
00205
00206 /* Link function for EEPROM peripheral over
I2C */
00207 void                      EEPROM_I2C_IO_Init(
void);
00208 HAL_StatusTypeDef         EEPROM_I2C_IO_Writ
eData(uint16_t DevAddress, uint16_t MemAddress, ui
```

```
nt8_t* pBuffer, uint32_t BufferSize);
00209 HAL_StatusTypeDef         EEPROM_I2C_IO_Read
Data(uint16_t DevAddress, uint16_t MemAddress, uin
t8_t* pBuffer, uint32_t BufferSize);
00210 HAL_StatusTypeDef         EEPROM_I2C_IO_IsDe
viceReady(uint16_t DevAddress, uint32_t Trials);
00211
00212 /* Link function for Temperature Sensor peri
pheral */
00213 void                      TSENSOR_IO_Init(vo
id);
00214 void                      TSENSOR_IO_Write(u
int16_t DevAddress, uint8_t* pBuffer, uint8_t Writ
eAddr, uint16_t Length);
00215 void                      TSENSOR_IO_Read(ui
nt16_t DevAddress, uint8_t* pBuffer, uint8_t ReadA
ddr, uint16_t Length);
00216 uint16_t                  TSENSOR_IO_IsDevic
eReady(uint16_t DevAddress, uint32_t Trials);
00217
00218 /* Link function for Audio Codec peripheral
*/
00219 void                      AUDIO_IO_Init(void
);
00220 void                      AUDIO_IO_DeInit(vo
id);
00221 void                      AUDIO_IO_Write(uin
t16_t DevAddress, uint8_t Reg, uint8_t Value);
00222 uint8_t                   AUDIO_IO_Read(uint
16_t DevAddress, uint8_t Reg);
00223 void                      AUDIO_IO_Delay(uin
t32_t delay);
00224 #endif /* HAL_I2C_MODULE_ENABLED */
00225
00226 #if defined(HAL_SPI_MODULE_ENABLED)
00227 /* SPIx bus function */
00228 static void               SPIx_Init(void);
```

```
00229 static void                 SPIx_Write(uint8_t
 Value);
00230 static uint32_t             SPIx_Read(void);
00231 static void                 SPIx_Error (void);
00232 static void                 SPIx_MspInit(SPI_H
andleTypeDef *hspi);
00233
00234 /* Link function for LCD peripheral over SPI
 */
00235 void                        LCD_IO_Init(void);
00236 void                        LCD_IO_WriteMultip
leData(uint8_t *pData, uint32_t Size);
00237 void                        LCD_IO_WriteReg(ui
nt8_t Reg);
00238 uint16_t                    LCD_IO_ReadData(ui
nt16_t Reg);
00239 void                        LCD_Delay (uint32_
t delay);
00240
00241 /* Link function for EEPROM peripheral over
SPI */
00242 void                        EEPROM_SPI_IO_Init(
void);
00243 void                        EEPROM_SPI_IO_Writ
eByte(uint8_t Data);
00244 uint8_t                     EEPROM_SPI_IO_Read
Byte(void);
00245 HAL_StatusTypeDef           EEPROM_SPI_IO_Writ
eData(uint16_t MemAddress, uint8_t* pBuffer, uint3
2_t BufferSize);
00246 HAL_StatusTypeDef           EEPROM_SPI_IO_Read
Data(uint16_t MemAddress, uint8_t* pBuffer, uint32
_t BufferSize);
00247 HAL_StatusTypeDef           EEPROM_SPI_IO_Wait
EepromStandbyState(void);
00248
00249 /* Link functions for SD Card peripheral ove
```

```
r SPI */
00250 void                        SD_IO_Init(void);
00251 HAL_StatusTypeDef          SD_IO_WriteCmd(uin
t8_t Cmd, uint32_t Arg, uint8_t Crc, uint8_t Respo
nse);
00252 HAL_StatusTypeDef          SD_IO_WaitResponse
(uint8_t Response);
00253 void                        SD_IO_WriteDummy(v
oid);
00254 void                        SD_IO_WriteByte(ui
nt8_t Data);
00255 uint8_t                     SD_IO_ReadByte(void
);
00256 #endif /* HAL_SPI_MODULE_ENABLED */
00257
00258 /**
00259   * @}
00260   */
00261
00262 /** @addtogroup STM32303C_EVAL_Exported_Func
tions
00263   * @{
00264   */
00265
00266 /**
00267   * @brief  This method returns the STM32303
C EVAL BSP Driver revision
00268   * @retval version : 0xXYZR (8bits for each
 decimal, R for RC)
00269   */
00270 uint32_t BSP_GetVersion(void)
00271 {
00272   return __STM32303C_EVAL_BSP_VERSION;
00273 }
00274
00275 /**
00276   * @brief  Configures LED GPIO.
```

```
00277   * @param  Led Specifies the Led to be conf
igured.
00278   *   This parameter can be one of following
 parameters:
00279   *     @arg LED1
00280   *     @arg LED2
00281   *     @arg LED3
00282   *     @arg LED4
00283   * @retval None
00284   */
00285 void BSP_LED_Init(Led_TypeDef Led)
00286 {
00287   GPIO_InitTypeDef  GPIO_InitStruct;
00288
00289   /* Enable the GPIO_LED clock */
00290   LEDx_GPIO_CLK_ENABLE(Led);
00291
00292   /* Configure the GPIO_LED pin */
00293   GPIO_InitStruct.Pin = LED_PIN[Led];
00294   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP
;
00295   GPIO_InitStruct.Pull = GPIO_PULLUP;
00296   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HI
GH;
00297
00298   HAL_GPIO_Init(LED_PORT[Led], &GPIO_InitStr
uct);
00299
00300   HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[L
ed], GPIO_PIN_SET);
00301 }
00302
00303 /**
00304   * @brief  Turns selected LED On.
00305   * @param  Led Specifies the Led to be set
on.
00306   *   This parameter can be one of following
```

```
 parameters:
00307   *     @arg LED1
00308   *     @arg LED2
00309   *     @arg LED3
00310   *     @arg LED4
00311   * @retval None
00312   */
00313 void BSP_LED_On(Led_TypeDef Led)
00314 {
00315   HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[L
ed], GPIO_PIN_RESET);
00316 }
00317
00318 /**
00319   * @brief  Turns selected LED Off.
00320   * @param  Led Specifies the Led to be set
off.
00321   *   This parameter can be one of following
 parameters:
00322   *     @arg LED1
00323   *     @arg LED2
00324   *     @arg LED3
00325   *     @arg LED4
00326   * @retval None
00327   */
00328 void BSP_LED_Off(Led_TypeDef Led)
00329 {
00330   HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[L
ed], GPIO_PIN_SET);
00331 }
00332
00333 /**
00334   * @brief  Toggles the selected LED.
00335   * @param  Led Specifies the Led to be togg
led.
00336   *   This parameter can be one of following
 parameters:
```

```
00337   *     @arg LED1
00338   *     @arg LED2
00339   *     @arg LED3
00340   *     @arg LED4
00341   * @retval None
00342   */
00343 void BSP_LED_Toggle(Led_TypeDef Led)
00344 {
00345   HAL_GPIO_TogglePin(LED_PORT[Led], LED_PIN[
Led]);
00346 }
00347
00348 /**
00349   * @brief  Configures push button GPIO and
EXTI Line.
00350   * @param  Button Button to be configured.
00351   *   This parameter can be one of the follo
wing values:
00352   *     @arg BUTTON_KEY   : Key Push Button
00353   * @param  Button_Mode Button mode requeste
d.
00354   *   This parameter can be one of the follo
wing values:
00355   *     @arg BUTTON_MODE_GPIO: Button will b
e used as simple IO
00356   *     @arg BUTTON_MODE_EXTI: Button will b
e connected to EXTI line
00357   *                            with interrup
t generation capability
00358   * @retval None
00359   */
00360 void BSP_PB_Init(Button_TypeDef Button, Butt
onMode_TypeDef Button_Mode)
00361 {
00362   GPIO_InitTypeDef GPIO_InitStruct;
00363
00364   /* Enable the corresponding Push Button cl
```

```
ock */
00365   BUTTONx_GPIO_CLK_ENABLE(Button);
00366
00367   /* Configure Push Button pin as input */
00368   GPIO_InitStruct.Pin = BUTTON_PIN[Button];
00369   GPIO_InitStruct.Pull = GPIO_PULLDOWN;
00370   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HI
GH;
00371
00372   if (Button_Mode == BUTTON_MODE_GPIO)
00373   {
00374     /* Configure Button pin as input */
00375     GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
00376     HAL_GPIO_Init(BUTTON_PORT[Button], &GPIO
_InitStruct);
00377   }
00378
00379   if (Button_Mode == BUTTON_MODE_EXTI)
00380   {
00381     if(Button == BUTTON_KEY)
00382     {
00383       /* Configure Key Push Button pin as in
put with External interrupt, falling edge */
00384       GPIO_InitStruct.Mode = GPIO_MODE_IT_FA
LLING;
00385     }
00386     else
00387     {
00388       /* Configure Joystick Push Button pin
as input with External interrupt, rising edge */
00389       GPIO_InitStruct.Mode = GPIO_MODE_IT_RI
SING;
00390     }
00391     HAL_GPIO_Init(BUTTON_PORT[Button], &GPIO
_InitStruct);
00392
00393     /* Enable and set Button EXTI Interrupt
```

```
to the lowest priority */
00394     HAL_NVIC_SetPriority((IRQn_Type)(BUTTON_
IRQn[Button]), 0x0F, 0x00);
00395     HAL_NVIC_EnableIRQ((IRQn_Type)(BUTTON_IR
Qn[Button]));
00396   }
00397 }
00398
00399 /**
00400   * @brief  Returns the selected button stat
e.
00401   * @param  Button Button to be checked.
00402   *   This parameter can be one of the follo
wing values:
00403   *     @arg BUTTON_KEY: Key Push Button
00404   * @retval The Button GPIO pin value
00405   */
00406 uint32_t BSP_PB_GetState(Button_TypeDef Butt
on)
00407 {
00408   return HAL_GPIO_ReadPin(BUTTON_PORT[Button
], BUTTON_PIN[Button]);
00409 }
00410
00411 /**
00412   * @brief  Configures all button of the joy
stick in GPIO or EXTI modes.
00413   * @param  Joy_Mode Joystick mode.
00414   *    This parameter can be one of the foll
owing values:
00415   *      @arg  JOY_MODE_GPIO: Joystick pins
will be used as simple IOs
00416   *      @arg  JOY_MODE_EXTI: Joystick pins
will be connected to EXTI line
00417   *                                  with int
errupt generation capability
00418   * @retval HAL_OK: if all initializations a
```

```
re OK. Other value if error.
00419   */
00420 uint8_t BSP_JOY_Init(JOYMode_TypeDef Joy_Mod
e)
00421 {
00422   JOYState_TypeDef joykey;
00423   GPIO_InitTypeDef GPIO_InitStruct;
00424
00425   /* Initialized the Joystick. */
00426   for(joykey = JOY_SEL; joykey < (JOY_SEL+JO
Yn) ; joykey++)
00427   {
00428     /* Enable the JOY clock */
00429     JOYx_GPIO_CLK_ENABLE(joykey);
00430
00431     GPIO_InitStruct.Pin = JOY_PIN[joykey];
00432     GPIO_InitStruct.Pull = GPIO_PULLDOWN;
00433     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_
HIGH;
00434
00435     if (Joy_Mode == JOY_MODE_GPIO)
00436     {
00437       /* Configure Joy pin as input */
00438       GPIO_InitStruct.Mode = GPIO_MODE_INPUT
;
00439       HAL_GPIO_Init(JOY_PORT[joykey], &GPIO_
InitStruct);
00440     }
00441
00442     if (Joy_Mode == JOY_MODE_EXTI)
00443     {
00444       /* Configure Joy pin as input with Ext
ernal interrupt */
00445       GPIO_InitStruct.Mode = GPIO_MODE_IT_RI
SING;
00446       HAL_GPIO_Init(JOY_PORT[joykey], &GPIO_
InitStruct);
```

```
00447
00448       /* Enable and set Joy EXTI Interrupt t
o the lowest priority */
00449       HAL_NVIC_SetPriority((IRQn_Type)(JOY_I
RQn[joykey]), 0x0F, 0x00);
00450       HAL_NVIC_EnableIRQ((IRQn_Type)(JOY_IRQn
[joykey]));
00451     }
00452   }
00453
00454   return HAL_OK;
00455 }
00456
00457 /**
00458   * @brief  Returns the current joystick sta
tus.
00459   * @retval Code of the joystick key pressed
00460   *          This code can be one of the fol
lowing values:
00461   *            @arg  JOY_NONE
00462   *            @arg  JOY_SEL
00463   *            @arg  JOY_DOWN
00464   *            @arg  JOY_LEFT
00465   *            @arg  JOY_RIGHT
00466   *            @arg  JOY_UP
00467   *            @arg  JOY_NONE
00468   */
00469 JOYState_TypeDef BSP_JOY_GetState(void)
00470 {
00471   JOYState_TypeDef joykey;
00472
00473   for(joykey = JOY_SEL; joykey < (JOY_SEL+JO
YN) ; joykey++)
00474   {
00475     if(HAL_GPIO_ReadPin(JOY_PORT[joykey], JO
Y_PIN[joykey]) == GPIO_PIN_SET)
00476     {
```

```
00477       /* Return Code Joystick key pressed */
00478       return joykey;
00479     }
00480   }
00481
00482   /* No Joystick key pressed */
00483   return JOY_NONE;
00484 }
00485
00486 #if defined(HAL_UART_MODULE_ENABLED)
00487 /**
00488   * @brief  Configures COM port.
00489   * @param  COM Specifies the COM port to be
 configured.
00490   *   This parameter can be one of following
 parameters:
00491   *     @arg COM1
00492   * @param  huart pointer to a UART_HandleTy
peDef structure that
00493   *   contains the configuration information
 for the specified UART peripheral.
00494   * @retval None
00495   */
00496 void BSP_COM_Init(COM_TypeDef COM, UART_Hand
leTypeDef* huart)
00497 {
00498   GPIO_InitTypeDef GPIO_InitStruct;
00499
00500   /* Enable GPIO clock */
00501   COMx_TX_GPIO_CLK_ENABLE(COM);
00502   COMx_RX_GPIO_CLK_ENABLE(COM);
00503
00504   /* Enable USART clock */
00505   COMx_CLK_ENABLE(COM);
00506
00507   /* Configure USART Tx as alternate functio
n push-pull */
```

```
  GPIO_InitStruct.Pin = COM_TX_PIN[COM];
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  GPIO_InitStruct.Alternate = COM_TX_AF[COM];
  HAL_GPIO_Init(COM_TX_PORT[COM], &GPIO_InitStruct);

  /* Configure USART Rx as alternate function push-pull */
  GPIO_InitStruct.Pin = COM_RX_PIN[COM];
  GPIO_InitStruct.Alternate = COM_RX_AF[COM];
  HAL_GPIO_Init(COM_RX_PORT[COM], &GPIO_InitStruct);

  /* USART configuration */
  huart->Instance = COM_USART[COM];
  HAL_UART_Init(huart);
}
#endif /* HAL_UART_MODULE_ENABLED) */
/**
  * @}
  */

/** @addtogroup STM32303C_EVAL_BUS
  * @{
  */
/*****************************************************************************
                                BUS OPERATIONS
*****************************************************************************/
#if defined(HAL_I2C_MODULE_ENABLED)
/****************************** I2C Routine
```

```
s**********************************/
00537
00538 /**
00539   * @brief Initializes I2C MSP
00540   * @param hi2c I2C handle
00541   * @retval None
00542   */
00543 static void I2Cx_MspInit(I2C_HandleTypeDef *
hi2c)
00544 {
00545   GPIO_InitTypeDef  GPIO_InitStruct;
00546   RCC_PeriphCLKInitTypeDef  RCC_PeriphCLKIni
tStruct;
00547
00548   if (hi2c->Instance == EVAL_I2Cx)
00549   {
00550     /*##-1- Configure the Eval I2C clock sou
rce. The clock is derived from the SYSCLK #*/
00551     RCC_PeriphCLKInitStruct.PeriphClockSelec
tion = RCC_PERIPHCLK_I2C2;
00552     RCC_PeriphCLKInitStruct.I2c2ClockSelecti
on = RCC_I2C2CLKSOURCE_SYSCLK;
00553     HAL_RCCEx_PeriphCLKConfig(&RCC_PeriphCLK
InitStruct);
00554
00555     /*##-2- Configure the GPIOs ############
####################################*/
00556
00557     /* Enable GPIO clock */
00558     EVAL_I2Cx_SDA_GPIO_CLK_ENABLE();
00559     EVAL_I2Cx_SCL_GPIO_CLK_ENABLE();
00560
00561     /* Configure I2C Tx as alternate functio
n  */
00562     GPIO_InitStruct.Pin       = EVAL_I2Cx_SC
L_PIN;
00563     GPIO_InitStruct.Mode      = GPIO_MODE_AF
```

```
_OD;
00564     GPIO_InitStruct.Pull      = GPIO_NOPULL;
00565     GPIO_InitStruct.Speed     = GPIO_SPEED_F
REQ_HIGH;
00566     GPIO_InitStruct.Alternate = EVAL_I2Cx_SC
L_SDA_AF;
00567     HAL_GPIO_Init(EVAL_I2Cx_SCL_GPIO_PORT, &
GPIO_InitStruct);
00568
00569     /* Configure I2C Rx as alternate functio
n  */
00570     GPIO_InitStruct.Pin = EVAL_I2Cx_SDA_PIN;
00571     HAL_GPIO_Init(EVAL_I2Cx_SDA_GPIO_PORT, &
GPIO_InitStruct);
00572
00573
00574     /*##-3- Configure the Eval I2Cx peripher
al ######################################*/
00575     /* Enable Eval_I2Cx clock */
00576     EVAL_I2Cx_CLK_ENABLE();
00577
00578     /* Force the I2C Periheral Clock Reset */

00579     EVAL_I2Cx_FORCE_RESET();
00580
00581     /* Release the I2C Periheral Clock Reset
 */
00582     EVAL_I2Cx_RELEASE_RESET();
00583
00584     /* Enable and set Eval I2Cx Interrupt to
 the lowest priority */
00585     HAL_NVIC_SetPriority(EVAL_I2Cx_EV_IRQn,
0x0F, 0);
00586     HAL_NVIC_EnableIRQ(EVAL_I2Cx_EV_IRQn);
00587
00588     /* Enable and set Eval I2Cx Interrupt to
 the lowest priority */
```

```
00589     HAL_NVIC_SetPriority(EVAL_I2Cx_ER_IRQn,
0x0F, 0);
00590     HAL_NVIC_EnableIRQ(EVAL_I2Cx_ER_IRQn);
00591   }
00592 }
00593
00594 /**
00595   * @brief Eval I2Cx Bus initialization
00596   * @retval None
00597   */
00598 static void I2Cx_Init(void)
00599 {
00600   if(HAL_I2C_GetState(&heval_I2c) == HAL_I2C
_STATE_RESET)
00601   {
00602     heval_I2c.Instance              = EVAL_I
2Cx;
00603     heval_I2c.Init.Timing           = EVAL_I
2Cx_TIMING;
00604     heval_I2c.Init.OwnAddress1      = 0;
00605     heval_I2c.Init.AddressingMode   = I2C_AD
DRESSINGMODE_7BIT;
00606     heval_I2c.Init.DualAddressMode  = I2C_DU
ALADDRESS_DISABLE;
00607     heval_I2c.Init.OwnAddress2      = 0;
00608     heval_I2c.Init.GeneralCallMode  = I2C_GE
NERALCALL_DISABLE;
00609     heval_I2c.Init.NoStretchMode    = I2C_NO
STRETCH_DISABLE;
00610
00611     /* Init the I2C */
00612     I2Cx_MspInit(&heval_I2c);
00613     HAL_I2C_Init(&heval_I2c);
00614   }
00615 }
00616
00617 /**
```

```
00618   * @brief  Write a value in a register of t
he device through BUS.
00619   * @param  Addr Device address on BUS Bus.

00620   * @param  Reg The target register address
to write
00621   * @param  RegSize The target register size
 (can be 8BIT or 16BIT)
00622   * @param  Value The target register value
to be written
00623   * @retval None
00624   */
00625 static void I2Cx_WriteData(uint16_t Addr, ui
nt8_t Reg, uint16_t RegSize, uint8_t Value)
00626   {
00627   HAL_StatusTypeDef status = HAL_OK;
00628
00629   status = HAL_I2C_Mem_Write(&heval_I2c, Add
r, (uint16_t)Reg, RegSize, &Value, 1, I2cxTimeout)
;
00630
00631   /* Check the communication status */
00632   if(status != HAL_OK)
00633   {
00634     /* Re-Initiaize the BUS */
00635     I2Cx_Error();
00636   }
00637 }
00638
00639 /**
00640   * @brief  Write a value in a register of t
he device through BUS.
00641   * @param  Addr Device address on BUS Bus.

00642   * @param  Reg The target register address
to write
00643   * @param  RegSize The target register size
```

```
 (can be 8BIT or 16BIT)
00644   * @param  pBuffer The target register valu
e to be written
00645   * @param  Length buffer size to be written
00646   * @retval None
00647   */
00648 static HAL_StatusTypeDef I2Cx_WriteBuffer(ui
nt16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_
t *pBuffer, uint16_t Length)
00649 {
00650   HAL_StatusTypeDef status = HAL_OK;
00651
00652   status = HAL_I2C_Mem_Write(&heval_I2c, Add
r, (uint16_t)Reg, RegSize, pBuffer, Length, I2cxTi
meout);
00653
00654 /* Check the communication status */
00655   if(status != HAL_OK)
00656   {
00657     /* Re-Initiaize the BUS */
00658     I2Cx_Error();
00659   }
00660   return status;
00661 }
00662
00663 /**
00664   * @brief  Read a register of the device th
rough BUS
00665   * @param  Addr Device address on BUS
00666   * @param  Reg The target register address
to read
00667   * @param  RegSize The target register size
 (can be 8BIT or 16BIT)
00668   * @retval read register value
00669   */
00670 static uint8_t I2Cx_ReadData(uint16_t Addr,
uint8_t Reg, uint16_t RegSize)
```

```
00671 {
00672   HAL_StatusTypeDef status = HAL_OK;
00673   uint8_t value = 0;
00674 
00675   status = HAL_I2C_Mem_Read(&heval_I2c, Addr
, Reg, RegSize, &value, 1, I2cxTimeout);
00676 
00677   /* Check the communication status */
00678   if(status != HAL_OK)
00679   {
00680     /* Re-Initiaize the BUS */
00681     I2Cx_Error();
00682 
00683   }
00684   return value;
00685 }
00686 
00687 /**
00688   * @brief  Reads multiple data on the BUS.
00689   * @param  Addr I2C Address
00690   * @param  Reg Reg Address
00691   * @param  RegSize The target register size
 (can be 8BIT or 16BIT)
00692   * @param  pBuffer pointer to read data buf
fer
00693   * @param  Length length of the data
00694   * @retval 0 if no problems to read multipl
e data
00695   */
00696 static HAL_StatusTypeDef I2Cx_ReadBuffer(uin
t16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t
 *pBuffer, uint16_t Length)
00697 {
00698   HAL_StatusTypeDef status = HAL_OK;
00699 
00700   status = HAL_I2C_Mem_Read(&heval_I2c, Addr
, (uint16_t)Reg, RegSize, pBuffer, Length, I2cxTim
```

```
eout);
00701
00702 /* Check the communication status */
00703   if(status != HAL_OK)
00704   {
00705     /* Re-Initiaize the BUS */
00706     I2Cx_Error();
00707   }
00708   return status;
00709 }
00710
00711 /**
00712 * @brief  Checks if target device is ready f
or communication.
00713 * @note   This function is used with Memory
devices
00714 * @param  DevAddress Target device address
00715 * @param  Trials Number of trials
00716 * @retval HAL status
00717 */
00718 static HAL_StatusTypeDef I2Cx_IsDeviceReady(
uint16_t DevAddress, uint32_t Trials)
00719 {
00720   return (HAL_I2C_IsDeviceReady(&heval_I2c,
DevAddress, Trials, I2cxTimeout));
00721 }
00722
00723
00724 /**
00725   * @brief Eval I2Cx error treatment function

00726   * @retval None
00727   */
00728 static void I2Cx_Error (void)
00729 {
00730   /* De-initialize the I2C communication BUS
 */
```

```
00731   HAL_I2C_DeInit(&heval_I2c);
00732
00733   /* Re- Initiaize the I2C communication BUS
 */
00734   I2Cx_Init();
00735 }
00736
00737 #endif /*HAL_I2C_MODULE_ENABLED*/
00738
00739 #if defined(HAL_SPI_MODULE_ENABLED)
00740 /****************************** SPI Routine
s ******************************/
00741 /**
00742   * @brief Initializes SPI MSP.
00743   * @param hspi SPI handle
00744   * @retval None
00745   */
00746 static void SPIx_MspInit(SPI_HandleTypeDef *
hspi)
00747 {
00748   GPIO_InitTypeDef  GPIO_InitStruct;
00749
00750   /*** Configure the GPIOs ***/
00751   /* Enable GPIO clock */
00752   EVAL_SPIx_SCK_GPIO_CLK_ENABLE();
00753   EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE();
00754
00755   /* configure SPI SCK */
00756   GPIO_InitStruct.Pin = EVAL_SPIx_SCK_PIN;
00757   GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00758   GPIO_InitStruct.Pull  = GPIO_PULLDOWN;
00759   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HI
GH;
00760   GPIO_InitStruct.Alternate = EVAL_SPIx_SCK_
AF;
00761   HAL_GPIO_Init(EVAL_SPIx_SCK_GPIO_PORT, &GP
IO_InitStruct);
```

```
00762
00763   /* configure SPI MISO and MOSI */
00764   GPIO_InitStruct.Pin = (EVAL_SPIx_MISO_PIN
| EVAL_SPIx_MOSI_PIN);
00765   GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00766   GPIO_InitStruct.Pull  = GPIO_NOPULL;
00767   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HI
GH;
00768   GPIO_InitStruct.Alternate = EVAL_SPIx_MISO
_MOSI_AF;
00769   HAL_GPIO_Init(EVAL_SPIx_MISO_MOSI_GPIO_PORT
, &GPIO_InitStruct);
00770
00771   /*** Configure the SPI peripheral ***/
00772   /* Enable SPI clock */
00773   EVAL_SPIx_CLK_ENABLE();
00774 }
00775
00776 /**
00777   * @brief  Initializes SPI HAL.
00778   * @retval None
00779   */
00780 static void SPIx_Init(void)
00781 {
00782   if(HAL_SPI_GetState(&heval_Spi) == HAL_SPI
_STATE_RESET)
00783   {
00784     /* SPI Config */
00785     heval_Spi.Instance = EVAL_SPIx;
00786     /* SPI baudrate is set to 18 MHz (PCLK2/
SPI_BaudRatePrescaler = 36/2 = 18 MHz) */
00787     heval_Spi.Init.BaudRatePrescaler = SPI_B
AUDRATEPRESCALER_2;
00788     heval_Spi.Init.Direction = SPI_DIRECTION
_2LINES;
00789     heval_Spi.Init.CLKPhase = SPI_PHASE_1EDG
E;
```

```
00790     heval_Spi.Init.CLKPolarity = SPI_POLARIT
Y_LOW;
00791     heval_Spi.Init.CRCCalculation = SPI_CRCC
ALCULATION_DISABLE;
00792     heval_Spi.Init.CRCPolynomial = 7;
00793     heval_Spi.Init.DataSize = SPI_DATASIZE_8
BIT;
00794     heval_Spi.Init.FirstBit = SPI_FIRSTBIT_M
SB;
00795     heval_Spi.Init.NSS = SPI_NSS_SOFT;
00796     heval_Spi.Init.TIMode = SPI_TIMODE_DISAB
LE;
00797     heval_Spi.Init.Mode = SPI_MODE_MASTER;
00798
00799     SPIx_MspInit(&heval_Spi);
00800     HAL_SPI_Init(&heval_Spi);
00801   }
00802 }
00803
00804 /**
00805   * @brief SPI Read 4 bytes from device
00806   * @retval Read data
00807 */
00808 static uint32_t SPIx_Read(void)
00809 {
00810   HAL_StatusTypeDef status = HAL_OK;
00811   uint32_t readvalue = 0;
00812   uint32_t writevalue = 0xFFFFFFFF;
00813
00814   status = HAL_SPI_TransmitReceive(&heval_Spi
, (uint8_t*) &writevalue, (uint8_t*) &readvalue, 1
, SpixTimeout);
00815
00816   /* Check the communication status */
00817   if(status != HAL_OK)
00818   {
00819     /* Execute user timeout callback */
```

```
00820     SPIx_Error();
00821   }
00822
00823   return readvalue;
00824 }
00825
00826 /**
00827   * @brief SPI Write a byte to device
00828   * @param Value value to be written
00829   * @retval None
00830   */
00831 static void SPIx_Write(uint8_t Value)
00832 {
00833   HAL_StatusTypeDef status = HAL_OK;
00834
00835   status = HAL_SPI_Transmit(&heval_Spi, (uin
t8_t*) &Value, 1, SpixTimeout);
00836
00837   /* Check the communication status */
00838   if(status != HAL_OK)
00839   {
00840     /* Execute user timeout callback */
00841     SPIx_Error();
00842   }
00843 }
00844
00845 /**
00846   * @brief SPI error treatment function
00847   * @retval None
00848   */
00849 static void SPIx_Error (void)
00850 {
00851   /* De-initialize the SPI communication BUS
 */
00852   HAL_SPI_DeInit(&heval_Spi);
00853
00854   /* Re- Initiaize the SPI communication BUS
```

```
 */
00855   SPIx_Init();
00856 }
00857 /**
00858   * @}
00859   */
00860
00861 /** @defgroup STM32303C_EVAL_LINK_OPERATIONS
 Link Operation functions
00862   * @{
00863   */
00864
00865 /****************************** LINK LCD
******************************/
00866
00867 /**
00868   * @brief  Configures the LCD_SPI interface.

00869   * @retval None
00870   */
00871 void LCD_IO_Init(void)
00872 {
00873   GPIO_InitTypeDef GPIO_InitStruct;
00874
00875   /* Configure the LCD Control pins --------
-----------------------------------*/
00876   LCD_NCS_GPIO_CLK_ENABLE();
00877
00878   /* Configure NCS in Output Push-Pull mode
*/
00879   GPIO_InitStruct.Pin     = LCD_NCS_PIN;
00880   GPIO_InitStruct.Mode    = GPIO_MODE_OUTPUT
_PP;
00881   GPIO_InitStruct.Pull    = GPIO_NOPULL;
00882   GPIO_InitStruct.Speed   = GPIO_SPEED_FREQ_
HIGH;
00883   HAL_GPIO_Init(LCD_NCS_GPIO_PORT, &GPIO_Ini
```

```
tStruct);
00884
00885   /* Set or Reset the control line */
00886   LCD_CS_LOW();
00887   LCD_CS_HIGH();
00888
00889   SPIx_Init();
00890 }
00891
00892 /**
00893   * @brief  Write register value.
00894   * @param  pData Pointer on the register va
lue
00895   * @param  Size Size of byte to transmit to
 the register
00896   * @retval None
00897   */
00898 void LCD_IO_WriteMultipleData(uint8_t *pData
, uint32_t Size)
00899 {
00900   uint32_t counter = 0;
00901
00902   /* Reset LCD control line(/CS) and Send da
ta */
00903   LCD_CS_LOW();
00904
00905   /* Send Start Byte */
00906   SPIx_Write(START_BYTE | LCD_WRITE_REG);
00907
00908   for (counter = Size; counter != 0; counter
--)
00909   {
00910     /* Need to invert bytes for LCD*/
00911     SPIx_Write(*(pData+1));
00912     SPIx_Write(*pData);
00913     counter--;
00914     pData += 2;
```

```
00915   }
00916
00917   /* Reset LCD control line(/CS) and Send da
ta */
00918   LCD_CS_HIGH();
00919 }
00920
00921 /**
00922   * @brief  register address.
00923   * @param  Reg
00924   * @retval None
00925   */
00926 void LCD_IO_WriteReg(uint8_t Reg)
00927 {
00928   /* Reset LCD control line(/CS) and Send co
mmand */
00929   LCD_CS_LOW();
00930
00931   /* Send Start Byte */
00932   SPIx_Write(START_BYTE | SET_INDEX);
00933
00934   /* Write 16-bit Reg Index (High Byte is 0)
 */
00935   SPIx_Write(0x00);
00936   SPIx_Write(Reg);
00937
00938   /* Deselect : Chip Select high */
00939   LCD_CS_HIGH();
00940 }
00941
00942 /**
00943   * @brief  Read register value.
00944   * @param  Reg
00945   * @retval None
00946   */
00947 uint16_t LCD_IO_ReadData(uint16_t Reg)
00948 {
```

```
00949   uint32_t readvalue = 0;
00950
00951   /* Change BaudRate Prescaler 8 for Read */
00952   /* Mean SPI baudrate is set to 72/8 = 9 MH
z */
00953   heval_Spi.Instance->CR1 &= 0xFFC7;
00954   heval_Spi.Instance->CR1 |= SPI_BAUDRATEPRE
SCALER_8;
00955
00956   /* Send Reg value to Read */
00957   LCD_IO_WriteReg(Reg);
00958
00959   /* Reset LCD control line(/CS) and Send co
mmand */
00960   LCD_CS_LOW();
00961
00962   /* Send Start Byte */
00963   SPIx_Write(START_BYTE | LCD_READ_REG);
00964
00965   /* Read Upper Byte */
00966   SPIx_Write(0xFF);
00967   readvalue = SPIx_Read();
00968   readvalue = readvalue << 8;
00969   readvalue |= SPIx_Read();
00970
00971   /* Recover Baud Rate initial value */
00972   heval_Spi.Instance->CR1 &= 0xFFC7;
00973   heval_Spi.Instance->CR1 |= heval_Spi.Init.
BaudRatePrescaler;
00974
00975   HAL_Delay(10);
00976
00977   /* Deselect : Chip Select high */
00978   LCD_CS_HIGH();
00979   return readvalue;
00980 }
00981
```

```
00982 /**
00983   * @brief  Wait for loop in ms.
00984   * @param  Delay in ms.
00985   * @retval None
00986   */
00987 void LCD_Delay (uint32_t Delay)
00988 {
00989   HAL_Delay(Delay);
00990 }
00991
00992 /********************************* LINK SD Ca
rd *********************************/
00993
00994 /**
00995   * @brief  Initializes the SD Card and put
it into StandBy State (Ready for
00996   *         data transfer).
00997   * @retval None
00998   */
00999 void SD_IO_Init(void)
01000 {
01001   GPIO_InitTypeDef  GPIO_InitStruct;
01002   uint8_t counter;
01003
01004   /* SD_CS_GPIO and SD_DETECT_GPIO Periph cl
ock enable */
01005   SD_CS_GPIO_CLK_ENABLE();
01006   SD_DETECT_GPIO_CLK_ENABLE();
01007
01008   /* Configure SD_CS_PIN pin: SD Card CS pin
 */
01009   GPIO_InitStruct.Pin = SD_CS_PIN;
01010   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP
;
01011   GPIO_InitStruct.Pull = GPIO_PULLUP;
01012   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HI
GH;
```

```
01013   HAL_GPIO_Init(SD_CS_GPIO_PORT, &GPIO_InitS
truct);
01014
01015   /* Configure SD_DETECT_PIN pin: SD Card de
tect pin */
01016   GPIO_InitStruct.Pin = SD_DETECT_PIN;
01017   GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING
_FALLING;
01018   GPIO_InitStruct.Pull = GPIO_PULLUP;
01019   HAL_GPIO_Init(SD_DETECT_GPIO_PORT, &GPIO_I
nitStruct);
01020
01021   /* Enable and set SD EXTI Interrupt to the
 lowest priority */
01022   HAL_NVIC_SetPriority(SD_DETECT_EXTI_IRQn,
0x0F, 0);
01023   HAL_NVIC_EnableIRQ(SD_DETECT_EXTI_IRQn);
01024
01025   /*------------Put SD in SPI mode----------
----*/
01026   /* SD SPI Config */
01027   SPIx_Init();
01028
01029   /* SD chip select high */
01030   SD_CS_HIGH();
01031
01032   /* Send dummy byte 0xFF, 10 times with CS
high */
01033   /* Rise CS and MOSI for 80 clocks cycles */

01034   for (counter = 0; counter <= 9; counter++)
01035   {
01036     /* Send dummy byte 0xFF */
01037     SD_IO_WriteByte(SD_DUMMY_BYTE);
01038   }
01039 }
01040
```

```
01041 /**
01042   * @brief  Write a byte on the SD.
01043   * @param  Data byte to send.
01044   * @retval None
01045   */
01046 void SD_IO_WriteByte(uint8_t Data)
01047 {
01048   /* Send the byte */
01049   SPIx_Write(Data);
01050 }
01051
01052 /**
01053   * @brief  Read a byte from the SD.
01054   * @retval The received byte.
01055   */
01056 uint8_t SD_IO_ReadByte(void)
01057 {
01058   uint8_t data = 0;
01059
01060   /* Get the received data */
01061   data = SPIx_Read();
01062
01063   /* Return the shifted data */
01064   return data;
01065 }
01066
01067 /**
01068   * @brief  Send 5 bytes command to the SD c
ard and get response
01069   * @param  Cmd The user expected command to
 send to SD card.
01070   * @param  Arg The command argument.
01071   * @param  Crc The CRC.
01072   * @param  Response Expected response from
the SD card
01073   * @retval  HAL_StatusTypeDef HAL Status
01074   */
```

```
01075 HAL_StatusTypeDef SD_IO_WriteCmd(uint8_t Cmd
, uint32_t Arg, uint8_t Crc, uint8_t Response)
01076 {
01077   uint32_t counter = 0x00;
01078   uint8_t frame[6];
01079
01080   /* Prepare Frame to send */
01081   frame[0] = (Cmd | 0x40); /* Construct byte
 1 */
01082   frame[1] = (uint8_t)(Arg >> 24); /* Constr
uct byte 2 */
01083   frame[2] = (uint8_t)(Arg >> 16); /* Constr
uct byte 3 */
01084   frame[3] = (uint8_t)(Arg >> 8); /* Constru
ct byte 4 */
01085   frame[4] = (uint8_t)(Arg); /* Construct by
te 5 */
01086   frame[5] = (Crc); /* Construct CRC: byte 6
 */
01087
01088   /* SD chip select low */
01089   SD_CS_LOW();
01090
01091   /* Send Frame */
01092   for (counter = 0; counter < 6; counter++)
01093   {
01094     SD_IO_WriteByte(frame[counter]); /* Send
 the Cmd bytes */
01095   }
01096
01097   if(Response != SD_NO_RESPONSE_EXPECTED)
01098   {
01099     return SD_IO_WaitResponse(Response);
01100   }
01101
01102   return HAL_OK;
01103 }
```

```
01104
01105 /**
01106   * @brief  Wait response from the SD card
01107   * @param  Response Expected response from
the SD card
01108   * @retval  HAL_StatusTypeDef HAL Status
01109   */
01110 HAL_StatusTypeDef SD_IO_WaitResponse(uint8_t
 Response)
01111 {
01112   uint32_t timeout = 0xFFF;
01113
01114   /* Check if response is got or a timeout i
s happen */
01115   while ((SD_IO_ReadByte() != Response) && t
imeout)
01116   {
01117     timeout--;
01118   }
01119
01120   if (timeout == 0)
01121   {
01122     /* After time out */
01123     return HAL_TIMEOUT;
01124   }
01125   else
01126   {
01127     /* Right response got */
01128     return HAL_OK;
01129   }
01130 }
01131
01132 /**
01133   * @brief  Send dummy byte with CS High
01134   * @retval None
01135   */
01136 void SD_IO_WriteDummy(void)
```

```
01137 {
01138   /* SD chip select high */
01139   SD_CS_HIGH();
01140
01141   /* Send Dummy byte 0xFF */
01142   SD_IO_WriteByte(SD_DUMMY_BYTE);
01143 }
01144
01145 /******************************** LINK EEPRO
M SPI ********************************/
01146
01147 /**
01148   * @brief  Initializes the EEPROM SPI and p
ut it into StandBy State (Ready for
01149   *         data transfer).
01150   * @retval None
01151   */
01152 void EEPROM_SPI_IO_Init(void)
01153 {
01154   GPIO_InitTypeDef  GPIO_InitStruct;
01155
01156   /* EEPROM_CS_GPIO Periph clock enable */
01157   EEPROM_CS_GPIO_CLK_ENABLE();
01158
01159   /* Configure EEPROM_CS_PIN pin: EEPROM SPI
 CS pin */
01160   GPIO_InitStruct.Pin = EEPROM_CS_PIN;
01161   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP
;
01162   GPIO_InitStruct.Pull = GPIO_PULLUP;
01163   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HI
GH;
01164   HAL_GPIO_Init(EEPROM_CS_GPIO_PORT, &GPIO_I
nitStruct);
01165
01166   /*------------Put EEPROM in SPI mode------
--------*/
```

```
01167   /* EEPROM SPI Config */
01168   SPIx_Init();
01169
01170   /* EEPROM chip select high */
01171   EEPROM_CS_HIGH();
01172 }
01173
01174 /**
01175   * @brief  Write a byte on the EEPROM.
01176   * @param  Data byte to send.
01177   * @retval None
01178   */
01179 void EEPROM_SPI_IO_WriteByte(uint8_t Data)
01180 {
01181   /* Send the byte */
01182   SPIx_Write(Data);
01183 }
01184
01185 /**
01186   * @brief  Read a byte from the EEPROM.
01187   * @retval uint8_t (The received byte).
01188   */
01189 uint8_t EEPROM_SPI_IO_ReadByte(void)
01190 {
01191   uint8_t data = 0;
01192
01193   /* Get the received data */
01194   data = SPIx_Read();
01195
01196   /* Return the shifted data */
01197   return data;
01198 }
01199
01200 /**
01201   * @brief  Write data to SPI EEPROM driver
01202   * @param  MemAddress Internal memory addre
ss
```

```
01203   * @param  pBuffer Pointer to data buffer
01204   * @param  BufferSize Amount of data to be
read
01205   * @retval HAL_StatusTypeDef HAL Status
01206   */
01207 HAL_StatusTypeDef EEPROM_SPI_IO_WriteData(ui
nt16_t MemAddress, uint8_t* pBuffer, uint32_t Buff
erSize)
01208 {
01209   /*!< Enable the write access to the EEPROM
 */
01210   /*!< Select the EEPROM: Chip Select low */
01211   EEPROM_CS_LOW();
01212
01213   /*!< Send "Write Enable" instruction */
01214   SPIx_Write(EEPROM_CMD_WREN);
01215
01216   /*!< Deselect the EEPROM: Chip Select high
 */
01217   EEPROM_CS_HIGH();
01218
01219   /*!< Select the EEPROM: Chip Select low */
01220   EEPROM_CS_LOW();
01221
01222   /*!< Send "Write to Memory " instruction */

01223   /* Send the byte */
01224   SPIx_Write(EEPROM_CMD_WRITE);
01225
01226   /*!< Send MemAddress high nibble address b
yte to write to */
01227   SPIx_Write((MemAddress & 0xFF0000) >> 16);
01228
01229   /*!< Send MemAddress medium nibble address
 byte to write to */
01230   SPIx_Write((MemAddress & 0xFF00) >> 8);
01231
```

```
01232   /*!< Send MemAddress low nibble address by
te to write to */
01233   SPIx_Write(MemAddress & 0xFF);
01234
01235   /*!< while there is data to be written on
the EEPROM */
01236   while ((BufferSize)--)
01237   {
01238     /*!< Send the current byte */
01239     SPIx_Write(*pBuffer);
01240     /*!< Point on the next byte to be writte
n */
01241     pBuffer++;
01242   }
01243
01244   /*!< Deselect the EEPROM: Chip Select high
 */
01245   EEPROM_CS_HIGH();
01246
01247   /*!< Wait the end of EEPROM writing */
01248   EEPROM_SPI_IO_WaitEepromStandbyState();
01249
01250   /*!< Disable the write access to the EEROM
 */
01251   EEPROM_CS_LOW();
01252
01253   /*!< Send "Write Disable" instruction */
01254   SPIx_Write(EEPROM_CMD_WRDI);
01255
01256   /*!< Deselect the EEPROM: Chip Select high
 */
01257   EEPROM_CS_HIGH();
01258
01259   return HAL_OK;
01260 }
01261
01262 /**
```

```
01263   * @brief  Read data from SPI EEPROM driver
01264   * @param  MemAddress Internal memory addre
ss
01265   * @param  pBuffer Pointer to data buffer
01266   * @param  BufferSize Amount of data to be
read
01267   * @retval HAL_StatusTypeDef HAL Status
01268   */
01269 HAL_StatusTypeDef EEPROM_SPI_IO_ReadData(uin
t16_t MemAddress, uint8_t* pBuffer, uint32_t Buffe
rSize)
01270 {
01271   /*!< Select the EEPROM: Chip Select low */
01272   EEPROM_CS_LOW();
01273
01274   /*!< Send "Write to Memory " instruction */

01275   SPIx_Write(EEPROM_CMD_READ);
01276
01277   /*!< Send MemAddress high nibble address b
yte to write to */
01278   SPIx_Write((MemAddress & 0xFF0000) >> 16);
01279
01280   /*!< Send WriteAddr medium nibble address
byte to write to */
01281   SPIx_Write((MemAddress & 0xFF00) >> 8);
01282
01283   /*!< Send WriteAddr low nibble address byt
e to write to */
01284   SPIx_Write(MemAddress & 0xFF);
01285
01286   while ((BufferSize)--) /*!< while there is
 data to be read */
01287   {
01288     /*!< Read a byte from the EEPROM */
01289     *pBuffer = SPIx_Read();
01290     /*!< Point to the next location where th
```

```
e byte read will be saved */
01291     pBuffer++;
01292   }
01293
01294   /*!< Deselect the EEPROM: Chip Select high
 */
01295   EEPROM_CS_HIGH();
01296
01297   return HAL_OK;
01298 }
01299
01300 /**
01301   * @brief  Wait response from the SPI EEPROM

01302   * @retval HAL_StatusTypeDef HAL Status
01303   */
01304 HAL_StatusTypeDef EEPROM_SPI_IO_WaitEepromSt
andbyState(void)
01305 {
01306   uint32_t timeout = 0xFFFF;
01307   uint32_t eepromstatus;
01308
01309   /*!< Select the EEPROM: Chip Select low */
01310   EEPROM_CS_LOW();
01311
01312   /*!< Send "Read Status Register" instructi
on */
01313   SPIx_Write(EEPROM_CMD_RDSR);
01314
01315   /*!< Loop as long as the memory is busy wi
th a write cycle */
01316   do
01317   {
01318     /*!< Send a dummy byte to generate the c
lock needed by the EEPROM
01319     and put the value of the status register
 in EEPROM Status variable */
```

```
01320     eepromstatus = SPIx_Read();
01321     timeout --;
01322   }
01323   while (((eepromstatus & EEPROM_WIP_FLAG) =
= SET) && timeout); /* Write in progress */
01324
01325   /*!< Deselect the EEPROM: Chip Select high
 */
01326   EEPROM_CS_HIGH();
01327
01328   if ((eepromstatus & EEPROM_WIP_FLAG) != SE
T)
01329   {
01330     /* Right response got */
01331     return HAL_OK;
01332   }
01333   else
01334   {
01335     /* After time out */
01336     return HAL_TIMEOUT;
01337   }
01338 }
01339 #endif /* HAL_SPI_MODULE_ENABLED */
01340
01341 #if defined(HAL_I2C_MODULE_ENABLED)
01342 /****************************** LINK I2C
EEPROM *****************************/
01343 /**
01344   * @brief  Initializes peripherals used by
the I2C EEPROM driver.
01345   * @retval None
01346   */
01347 void EEPROM_I2C_IO_Init(void)
01348 {
01349   I2Cx_Init();
01350 }
01351
```

```
01352 /**
01353   * @brief  Write data to I2C EEPROM driver
01354   * @param  DevAddress Target device address
01355   * @param  MemAddress Internal memory addre
ss
01356   * @param  pBuffer Pointer to data buffer
01357   * @param  BufferSize Amount of data to be
sent
01358   * @retval HAL status
01359   */
01360 HAL_StatusTypeDef EEPROM_I2C_IO_WriteData(ui
nt16_t DevAddress, uint16_t MemAddress, uint8_t* p
Buffer, uint32_t BufferSize)
01361 {
01362   return (I2Cx_WriteBuffer(DevAddress, MemAd
dress, I2C_MEMADD_SIZE_16BIT, pBuffer, BufferSize)
);
01363 }
01364
01365 /**
01366   * @brief  Read data from I2C EEPROM driver
01367   * @param  DevAddress Target device address
01368   * @param  MemAddress Internal memory addre
ss
01369   * @param  pBuffer Pointer to data buffer
01370   * @param  BufferSize Amount of data to be
read
01371   * @retval HAL status
01372   */
01373 HAL_StatusTypeDef EEPROM_I2C_IO_ReadData(uin
t16_t DevAddress, uint16_t MemAddress, uint8_t* pB
uffer, uint32_t BufferSize)
01374 {
01375   return (I2Cx_ReadBuffer(DevAddress, MemAdd
ress, I2C_MEMADD_SIZE_16BIT, pBuffer, BufferSize))
;
01376 }
```

```
01377
01378 /**
01379 * @brief  Checks if target device is ready f
or communication.
01380 * @note   This function is used with Memory
devices
01381 * @param  DevAddress Target device address
01382 * @param  Trials Number of trials
01383 * @retval HAL status
01384 */
01385 HAL_StatusTypeDef EEPROM_I2C_IO_IsDeviceReady
(uint16_t DevAddress, uint32_t Trials)
01386 {
01387   return (I2Cx_IsDeviceReady(DevAddress, Tri
als));
01388 }
01389
01390 /****************************** LINK I2C
TEMPERATURE SENSOR *****************************/
01391 /**
01392   * @brief  Initializes peripherals used by
the I2C Temperature Sensor driver.
01393   * @retval None
01394   */
01395 void TSENSOR_IO_Init(void)
01396 {
01397   I2Cx_Init();
01398 }
01399
01400 /**
01401   * @brief  Writes one byte to the TSENSOR.
01402   * @param  DevAddress Target device address
01403   * @param  pBuffer Pointer to data buffer
01404   * @param  WriteAddr TSENSOR's internal add
ress to write to.
01405   * @param  Length Number of data to write
01406   * @retval None
```

```
01407   */
01408 void TSENSOR_IO_Write(uint16_t DevAddress, u
int8_t* pBuffer, uint8_t WriteAddr, uint16_t Lengt
h)
01409 {
01410   I2Cx_WriteBuffer(DevAddress, WriteAddr, I2
C_MEMADD_SIZE_8BIT, pBuffer, Length);
01411 }
01412
01413 /**
01414   * @brief  Reads one byte from the TSENSOR.
01415   * @param  DevAddress Target device address
01416   * @param  pBuffer pointer to the buffer th
at receives the data read from the TSENSOR.
01417   * @param  ReadAddr TSENSOR's internal addr
ess to read from.
01418   * @param  Length Number of data to read
01419   * @retval None
01420   */
01421 void TSENSOR_IO_Read(uint16_t DevAddress, ui
nt8_t* pBuffer, uint8_t ReadAddr, uint16_t Length)
01422 {
01423   I2Cx_ReadBuffer(DevAddress, ReadAddr, I2C_
MEMADD_SIZE_8BIT, pBuffer, Length);
01424 }
01425
01426 /**
01427 * @brief  Checks if Temperature Sensor is re
ady for communication.
01428   * @param  DevAddress Target device address
01429   * @param  Trials Number of trials
01430 * @retval HAL status
01431 */
01432 uint16_t TSENSOR_IO_IsDeviceReady(uint16_t D
evAddress, uint32_t Trials)
01433 {
01434   return (I2Cx_IsDeviceReady(DevAddress, Tri
```

```
als));
01435 }
01436
01437
01438 /********************************** LINK AUDI
O CODEC ***********************************/
01439 /**
01440   * @brief  Initializes peripherals used by
the Audio Codec driver.
01441   * @retval None
01442   */
01443 void AUDIO_IO_Init(void)
01444 {
01445   I2Cx_Init();
01446 }
01447
01448 /**
01449   * @brief  DeInitializes Audio low level.
01450   * @note   This function is intentionally k
ept empty, user should define it.
01451   */
01452 void AUDIO_IO_DeInit(void)
01453 {
01454
01455 }
01456
01457 /**
01458   * @brief  Writes a single data on the Audi
o Codec.
01459   * @param  DevAddress Target device address
01460   * @param  Reg Target Register address
01461   * @param  Value Data to be written
01462   * @retval None
01463   */
01464 void AUDIO_IO_Write(uint16_t DevAddress, uin
t8_t Reg, uint8_t Value)
01465 {
```

```
01466   I2Cx_WriteData(DevAddress, Reg, I2C_MEMADD
_SIZE_8BIT, Value);
01467 }
01468
01469 /**
01470   * @brief  Reads a single data from the Aud
io Codec.
01471   * @param  DevAddress Target device address
01472   * @param  Reg Target Register address
01473   * @retval Data to be read
01474   */
01475 uint8_t AUDIO_IO_Read(uint16_t DevAddress, u
int8_t Reg)
01476 {
01477   uint8_t value;
01478
01479   value = I2Cx_ReadData(DevAddress, Reg, I2C
_MEMADD_SIZE_8BIT);
01480
01481   return value;
01482 }
01483
01484 /**
01485   * @brief  Wait for loop in ms.
01486   * @param  Delay in ms.
01487   * @retval None
01488   */
01489 void AUDIO_IO_Delay(uint32_t Delay)
01490 {
01491   HAL_Delay(Delay);
01492 }
01493
01494 #endif /* HAL_I2C_MODULE_ENABLED */
01495
01496 /**
01497   * @}
01498   */
```

```
01499
01500 /**
01501   * @}
01502   */
01503
01504 /**
01505   * @}
01506   */
01507
01508 /**
01509   * @}
01510   */
01511
01512 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM32303C_EVAL BSP User Manual

stm32303c_eval_audio.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32303c_eval_audio.h
00004   * @author  MCD Application Team
00005   * @brief   This file contains all the functions prototypes for the
00006   *          stm32303c_eval_audio.c driver.
00007   ******************************************************************************
00008   * @attention
00009   *
00010   * <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>
00011   *
00012   * Redistribution and use in source and binary forms, with or without modification,
00013   * are permitted provided that the following conditions are met:
00014   *   1. Redistributions of source code must retain the above copyright notice,
00015   *      this list of conditions and the fol
```

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

```
00035   */
00036
00037 /* Define to prevent recursive inclusion ---
------------------------------------*/
00038 #ifndef __STM32303C_EVAL_AUDIO_H
00039 #define __STM32303C_EVAL_AUDIO_H
00040
00041 #ifdef __cplusplus
00042  extern "C" {
00043 #endif
00044
00045 /* Includes ------------------------------
------------------------------------*/
00046 /* Include AUDIO component driver */
00047 #include "../Components/cs42l52/cs42l52.h"

00048 #include "stm32303c_eval.h"
00049 /** @addtogroup BSP
00050   * @{
00051   */
00052
00053 /** @addtogroup STM32303C_EVAL
00054   * @{
00055   */
00056
00057 /** @addtogroup STM32303C_EVAL_AUDIO
00058   * @{
00059   */
00060
00061 /** @defgroup STM32303C_EVAL_AUDIO_Exported_
Types Exported Types
00062   * @{
00063   */
00064 typedef enum
00065 {
00066   AUDIO_OK       = 0x00,
00067   AUDIO_ERROR    = 0x01,
```

```
00068   AUDIO_TIMEOUT  = 0x02
00069
00070 }AUDIO_StatusTypeDef;
00071
00072 /**
00073   * @}
00074   */
00075
00076 /** @defgroup STM32303C_EVAL_AUDIO_Exported_
Constants Exported Constants
00077   * @{
00078   */
00079
00080 /* Audio Codec hardware I2C address */
00081 #define AUDIO_I2C_ADDRESS
    0x94
00082
00083 /*-----------------------------------------
-----------------------------------
00084              AUDIO OUT CONFIGURATION
00085   -----------------------------------------
-----------------------------------*/
00086
00087 /* I2S peripheral configuration defines */
00088 #define I2Sx
    SPI3
00089 #define I2Sx_CLK_ENABLE()
    __HAL_RCC_SPI3_CLK_ENABLE()
00090 #define I2Sx_CLK_DISABLE()
    __HAL_RCC_SPI3_CLK_DISABLE()
00091 #define I2Sx_FORCE_RESET()
    __HAL_RCC_SPI3_FORCE_RESET()
00092 #define I2Sx_RELEASE_RESET()
    __HAL_RCC_SPI3_RELEASE_RESET()
00093
00094 #define I2Sx_WS_PIN
    GPIO_PIN_4
```

```
00095 #define I2Sx_MCK_PIN
    GPIO_PIN_9
00096 #define I2Sx_SCK_PIN
    GPIO_PIN_10
00097 #define I2Sx_DIN_PIN
    GPIO_PIN_12
00098
00099 #define I2Sx_WS_GPIO_PORT
    GPIOA
00100 #define I2Sx_MCK_GPIO_PORT
    GPIOA
00101 #define I2Sx_SCK_DIN_GPIO_PORT
    GPIOC
00102 #define I2Sx_MCK_WS_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOA_CLK_ENABLE()
00103 #define I2Sx_MCK_WS_GPIO_CLK_DISABLE()
    __HAL_RCC_GPIOA_CLK_DISABLE()
00104 #define I2Sx_WS_AF
    GPIO_AF6_SPI3
00105 #define I2Sx_MCK_AF
    GPIO_AF5_SPI3
00106 #define I2Sx_SCK_DIN_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOC_CLK_ENABLE()
00107 #define I2Sx_SCK_DIN_GPIO_CLK_DISABLE()
    __HAL_RCC_GPIOC_CLK_DISABLE()
00108 #define I2Sx_SCK_DIN_AF
    GPIO_AF6_SPI3
00109
00110 /* I2S DMA Stream definitions */
00111 #define I2Sx_DMAx_CLK_ENABLE()
    __HAL_RCC_DMA2_CLK_ENABLE()
00112 #define I2Sx_DMAx_CLK_DISABLE()
    __HAL_RCC_DMA2_CLK_DISABLE()
00113 #define I2Sx_DMAx_CHANNEL
    DMA2_Channel2
00114 #define I2Sx_DMAx_IRQ
    DMA2_Channel2_IRQn
```

```
00115 #define I2Sx_DMAx_PERIPH_DATA_SIZE
    DMA_PDATAALIGN_HALFWORD
00116 #define I2Sx_DMAx_MEM_DATA_SIZE
    DMA_MDATAALIGN_HALFWORD
00117 #define DMA_MAX_SZE
    0xFFFF
00118
00119 /* Select the interrupt preemption priority
and subpriority for the DMA interrupt */
00120 #define AUDIO_OUT_IRQ_PREPRIO
    ((uint32_t)0x0E)   /* Select the preemption pr
iority level(0 is the highest) */
00121 #define AUDIO_OUT_IRQ_SUBPRIO
    0   /* Select the sub-priority level (0 is the
 highest) */
00122 /*-----------------------------------------
-----------------------------------*/
00123
00124 /*-----------------------------------------
-----------------------------------
00125                       OPTIONAL Configuration d
efines parameters
00126 -----------------------------------------
-----------------------------------*/
00127 #define AUDIODATA_SIZE        2   /* 16-bits
 audio data size */
00128
00129 #define DMA_MAX(_X_)          (((_X_) <= DMA
_MAX_SZE)? (_X_):DMA_MAX_SZE)
00130 /**
00131   * @}
00132   */
00133
00134 /** @defgroup STM32303C_EVAL_AUDIO_Exported_
Variables Exported Variables
00135   * @{
00136   */
```

```
00137
00138 /**
00139   * @}
00140   */
00141
00142 /** @defgroup STM32303C_EVAL_AUDIO_Exported_
Macros Exported Macros
00143   * @{
00144   */
00145 /**
00146   * @}
00147   */
00148
00149 /* Exported functions ----------------------
-----------------------------------*/
00150 /** @defgroup STM32303C_EVAL_AUDIO_Exported_
Functions Exported Functions
00151   * @{
00152   */
00153 uint8_t        BSP_AUDIO_OUT_Init(uint16_t O
utputDevice, uint8_t Volume, uint32_t AudioFreq);
00154 uint8_t        BSP_AUDIO_OUT_Play(uint16_t*
pBuffer, uint32_t Size);
00155 uint8_t        BSP_AUDIO_OUT_ChangeBuffer(ui
nt16_t *pData, uint16_t Size);
00156 uint8_t        BSP_AUDIO_OUT_Pause(void);
00157 uint8_t        BSP_AUDIO_OUT_Resume(void);
00158 uint8_t        BSP_AUDIO_OUT_Stop(uint32_t O
ption);
00159 uint8_t        BSP_AUDIO_OUT_SetVolume(uint8
_t Volume);
00160 uint8_t        BSP_AUDIO_OUT_SetFrequency(ui
nt32_t AudioFreq);
00161 uint8_t        BSP_AUDIO_OUT_SetMute(uint32_
t Command);
00162 uint8_t        BSP_AUDIO_OUT_SetOutputMode(u
int8_t Output);
```

```
00163
00164 /* User Callbacks: user has to implement the
se functions in his code if they are needed. */
00165 /* This function is called when the requeste
d data has been completely transferred.*/
00166 void     BSP_AUDIO_OUT_TransferComplete_Call
Back(void);
00167
00168 /* This function is called when half of the
requested buffer has been transferred. */
00169 void     BSP_AUDIO_OUT_HalfTransfer_CallBack(
void);
00170
00171 /* This function is called when an Interrupt
 due to transfer error on or peripheral
00172    error occurs. */
00173 void     BSP_AUDIO_OUT_Error_CallBack(void);
00174 /**
00175   * @}
00176   */
00177
00178 /**
00179   * @}
00180   */
00181
00182 /**
00183   * @}
00184   */
00185
00186 /**
00187   * @}
00188   */
00189
00190 #ifdef __cplusplus
00191 }
00192 #endif
00193
```

```
00194 #endif /* __STM32303C_EVAL_AUDIO_H */
00195
00196 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM32303C_EVAL BSP User Manual

stm32303c_eval_audio.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32303c_eval_audio.c
00004   * @author  MCD Application Team
00005   * @brief   This file provides the Audio driver for the STM32303C_EVAL
00006   *          evaluation board(MB1019).
00007   ******************************************************************************
00008   * @attention
00009   *
00010   * <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>
00011   *
00012   * Redistribution and use in source and binary forms, with or without modification,
00013   * are permitted provided that the following conditions are met:
00014   *   1. Redistributions of source code must retain the above copyright notice,
00015   *      this list of conditions and the fol
```

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

```
00035   */
00036
00037 /*=========================================
=================================
00038
 User NOTES
00039 How To use this driver:
00040 -----------------------
00041    + This driver supports STM32F30x devices
on STM32303C_EVAL Evaluation boards:
00042    + Call the function BSP_AUDIO_OUT_Init(
00043                                           OutputDe
vice: physical output mode (OUTPUT_DEVICE_SPEAKER,

00044
     OUTPUT_DEVICE_HEADPHONE, OUTPUT_DEVICE_AUTO o
r
00045
     OUTPUT_DEVICE_BOTH)
00046                                          Volume:
initial volume to be set (0 is min (mute), 100 is
max (100%)
00047                                          AudioFre
q: Audio frequency in Hz (8000, 16000, 22500, 3200
0 ...)
00048                                           this pa
rameter is relative to the audio file/stream type.
00049                                         )
00050       This function configures all the hardw
are required for the audio application (codec, I2C
, I2S,
00051       GPIOs, DMA and interrupt if needed). T
his function returns 0 if configuration is OK.
00052       if the returned value is different fro
m 0 or the function is stuck then the communicatio
n with
00053       the codec has failed (try to un-plug t
```

```
he power or reset device in this case).
00054       - OUTPUT_DEVICE_SPEAKER: only speaker
will be set as output for the audio stream.
00055       - OUTPUT_DEVICE_HEADPHONE: only headph
ones will be set as output for the audio stream.
00056       - OUTPUT_DEVICE_AUTO: Selection of out
put device is made through external switch (implem
ented
00057           into the audio jack on the evaluati
on board). When the Headphone is connected it is u
sed
00058           as output. When the headphone is di
sconnected from the audio jack, the output is
00059           automatically switched to Speaker.
00060       - OUTPUT_DEVICE_BOTH: both Speaker and
 Headphone are used as outputs for the audio stream

00061           at the same time.
00062    + Call the function BSP_AUDIO_OUT_Play(
00063                                       pBuffer: p
ointer to the audio data file address
00064                                       Size: size
 of the buffer to be sent in Bytes
00065                                     )
00066       to start playing (for the first time)
from the audio file/stream.
00067    + Call the function BSP_AUDIO_OUT_Pause()
 to pause playing
00068    + Call the function BSP_AUDIO_OUT_Resume(
) to resume playing.
00069        Note. After calling BSP_AUDIO_OUT_Pau
se() function for pause, only BSP_AUDIO_OUT_Resume
() should be called
00070           for resume (it is not allowed to c
all BSP_AUDIO_OUT_Play() in this case).
00071        Note. This function should be called
only when the audio file is played or paused (not
```

```
stopped).
00072    + For each mode, you may need to implemen
t the relative callback functions into your code.
00073       The Callback functions are named BSP_A
UDIO_OUT_XXX_CallBack() and only their prototypes
are declared in
00074       the stm32303c_eval_audio.h file. (refe
r to the example for more details on the callbacks
 implementations)
00075    + To Stop playing, to modify the volume l
evel or to mute, use the functions
00076        BSP_AUDIO_OUT_Stop(), BSP_AUDIO_OUT_S
etVolume(), BSP_AUDIO_OUT_SetFrequency(), BSP_AUDI
O_OUT_SetOutputMode() and BSP_AUDIO_OUT_SetMute().
00077    + The driver API and the callback functio
ns are at the end of the stm32303c_eval_audio.h fi
le.
00078
00079    Driver architecture:
00080    -------------------
00081    + This driver provide the High Audio Laye
r: consists of the function API exported in the st
m32303c_eval_audio.h file
00082        (BSP_AUDIO_OUT_Init(), BSP_AUDIO_OUT_
Play() ...)
00083 + This driver provide also the Media Access
Layer (MAL): which consists of functions allowing
to access the media containing/
00084        providing the audio file/stream. Thes
e functions are also included as local functions i
nto
00085        the stm32303c_eval_audio.c file (I2Sx
_MspInit() and I2Sx_Init())
00086
00087  Known Limitations:
00088 -------------------
00089    1- When using the Speaker, if the audio f
```

```
ile quality is not high enough, the speaker output
00090       may produce high and uncomfortable noi
se level. To avoid this issue, to use speaker
00091       output properly, try to increase audio
 file sampling rate (typically higher than 48KHz).
00092       This operation will lead to larger fil
e size.
00093    2- Communication with the audio codec (th
rough I2C) may be corrupted if it is interrupted b
y some
00094       user interrupt routines (in this case,
 interrupts could be disabled just before the star
t of
00095       communication then re-enabled when it
 is over). Note that this communication is only don
e at
00096     the configuration phase (BSP_AUDIO_OUT_I
nit() or BSP_AUDIO_OUT_Stop()) and when Volume con
trol modification is
00097       performed (BSP_AUDIO_OUT_SetVolume() o
r AUDIO_OUT_Mute() or BSP_AUDIO_OUT_SetOutputMode(
)).
00098       When the audio data is played, no comm
unication is required with the audio codec.
00099   3- Parsing of audio file is not implemente
d (in order to determine audio file properties: Mo
no/Stereo, Data size,
00100      File size, Audio Frequency, Audio Data
 header size ...). The configuration is fixed for t
he given audio file.
00101   4- Mono audio streaming is not supported (
in order to play mono audio streams, each data sho
uld be sent twice
00102      on the I2S or should be duplicated on t
he source buffer. Or convert the stream in stereo
 before playing).
00103    5- Supports only 16-bits audio data size.
```

```
00104 ==========================================
==================================*/
00105
00106 /* Includes ------------------------------
-----------------------------------*/
00107 #include "stm32303c_eval_audio.h"
00108
00109 /** @addtogroup BSP
00110   * @{
00111   */
00112
00113 /** @addtogroup STM32303C_EVAL
00114   * @{
00115   */
00116
00117 /** @defgroup STM32303C_EVAL_AUDIO STM32303C
_EVAL AUDIO
00118   * @brief       This file includes the low
layer audio driver available on STM32303C_EVAL
00119   * evaluation board(MB1019).
00120   * @{
00121   */
00122
00123 /** @defgroup STM32303C_EVAL_AUDIO_Private_T
ypes Private Types
00124   * @{
00125   */
00126 /**
00127   * @}
00128   */
00129
00130 /* Private defines ------------------------
-----------------------------------*/
00131 /** @defgroup STM32303C_EVAL_AUDIO_Private_C
onstants Private Constants
00132   * @{
00133   */
```

```
00134 /**
00135   * @}
00136   */
00137
00138 /* Private macros ------------------------
------------------------------------*/
00139 /** @defgroup STM32303C_EVAL_AUDIO_Private_M
acros Private Macros
00140   * @{
00141   */
00142 /**
00143   * @}
00144   */
00145
00146 /* Private variables ---------------------
------------------------------------*/
00147 /** @defgroup STM32303C_EVAL_AUDIO_Private_V
ariables Private Variables
00148   * @{
00149   */
00150
00151 /*### PLAY ###*/
00152 static AUDIO_DrvTypeDef           *pAudioDrv
 = NULL;
00153 I2S_HandleTypeDef                 hAudioOutI
2s;
00154
00155 /**
00156   * @}
00157   */
00158
00159 /* Private function prototypes -----------
------------------------------------*/
00160 /** @defgroup STM32303C_EVAL_AUDIO_Private_F
unctions Private Functions
00161   * @{
00162   */
```

```
00163 static void                I2Sx_MspInit(void
);
00164 static AUDIO_StatusTypeDef  I2Sx_Init(uint32
_t AudioFreq);
00165 /**
00166   * @}
00167   */
00168
00169 /* Exported functions ---------------------
-----------------------------------*/
00170 /** @addtogroup STM32303C_EVAL_AUDIO_Exporte
d_Functions
00171   * @{
00172   */
00173
00174 /**
00175   * @brief  Configure the audio peripherals.
00176   * @param  OutputDevice OUTPUT_DEVICE_SPEAK
ER, OUTPUT_DEVICE_HEADPHONE,
00177   *                         OUTPUT_DEVICE_BOTH
 or OUTPUT_DEVICE_AUTO .
00178   * @param  Volume Initial volume level (fro
m 0 (Mute) to 100 (Max))
00179   * @param  AudioFreq Audio frequency used t
o play the audio stream.
00180   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00181   */
00182 uint8_t BSP_AUDIO_OUT_Init(uint16_t OutputDe
vice, uint8_t Volume, uint32_t AudioFreq)
00183 {
00184   uint8_t ret = AUDIO_OK;
00185   uint32_t deviceid = 0x00;
00186
00187   if(pAudioDrv == NULL)
00188   {
00189     deviceid = cs42l52_drv.ReadID(AUDIO_I2C_
```

```
ADDRESS);
00190
00191     if((deviceid & CS42L52_ID_MASK) == CS42L
52_ID)
00192     {
00193               /* Initialize the audio driver
 structure */
00194       pAudioDrv = &cs42l52_drv;
00195       ret = AUDIO_OK;
00196     }
00197     else
00198     {
00199       ret = AUDIO_ERROR;
00200     }
00201   }
00202
00203   if(ret == AUDIO_OK)
00204   {
00205     if(pAudioDrv->Init(AUDIO_I2C_ADDRESS, Ou
tputDevice, Volume, AudioFreq) != 0)
00206     {
00207       ret = AUDIO_ERROR;
00208     }
00209     else
00210     {
00211       /* I2S data transfer preparation:
00212       Prepare the Media to be used for the a
udio transfer from memory to I2S peripheral */
00213       /* Configure the I2S peripheral */
00214       ret = I2Sx_Init(AudioFreq);
00215     }
00216   }
00217
00218   return ret;
00219 }
00220
00221 /**
```

```
00222   * @brief  Starts playing audio stream from
 a data buffer for a determined size.
00223   * @param  pBuffer Pointer to the buffer
00224   * @param  Size Number of audio data BYTES.
00225   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00226   */
00227 uint8_t BSP_AUDIO_OUT_Play(uint16_t* pBuffer
, uint32_t Size)
00228 {
00229   /* Call the audio Codec Play function */
00230   if (pAudioDrv->Play(AUDIO_I2C_ADDRESS, pBu
ffer, Size) != 0)
00231   {
00232     return AUDIO_ERROR;
00233   }
00234   else
00235   {
00236     /* Update the Media layer and enable it
for play */
00237     return (HAL_I2S_Transmit_DMA(&hAudioOutI
2s, pBuffer, DMA_MAX(Size)));
00238   }
00239 }
00240
00241 /**
00242   * @brief Sends n-Bytes on the I2S interfac
e.
00243   * @param pData pointer on data address
00244   * @param Size number of data to be written
00245   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00246   */
00247 uint8_t BSP_AUDIO_OUT_ChangeBuffer(uint16_t
*pData, uint16_t Size)
00248 {
00249    return (HAL_I2S_Transmit_DMA(&hAudioOutI2s
```

```
, pData, Size));
00250 }
00251
00252 /**
00253   * @brief  This function Pauses the audio f
ile stream. In case of using DMA, the DMA Pause fe
ature is used.
00254   * @note When calling BSP_AUDIO_OUT_Pause()
 function for pause, only
00255   *          BSP_AUDIO_OUT_Resume() function
 should be called for resume (use of BSP_AUDIO_OUT
_Play()
00256   *          function for resume could lead
to unexpected behavior).
00257   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00258   */
00259 uint8_t BSP_AUDIO_OUT_Pause(void)
00260 {
00261   /* Call the Audio Codec Pause/Resume funct
ion */
00262   if (pAudioDrv->Pause(AUDIO_I2C_ADDRESS) !=
 0)
00263   {
00264     return AUDIO_ERROR;
00265   }
00266   else
00267   {
00268     /* Call the Media layer pause function */

00269     return (HAL_I2S_DMAPause(&hAudioOutI2s))
;
00270   }
00271 }
00272
00273 /**
00274   * @brief  This function  Resumes the audio
```

```
 file stream.
00275   * @note When calling BSP_AUDIO_OUT_Pause()
 function for pause, only
00276   *          BSP_AUDIO_OUT_Resume() function
 should be called for resume (use of BSP_AUDIO_OUT
_Play()
00277   *          function for resume could lead
to unexpected behavior).
00278   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00279   */
00280 uint8_t BSP_AUDIO_OUT_Resume(void)
00281 {
00282   /* Call the Audio Codec Pause/Resume funct
ion */
00283   if(pAudioDrv->Resume(AUDIO_I2C_ADDRESS) !=
 0)
00284   {
00285     return AUDIO_ERROR;
00286   }
00287   else
00288   {
00289     /* Call the Media layer resume function
*/
00290     return (HAL_I2S_DMAResume(&hAudioOutI2s)
);
00291   }
00292 }
00293
00294 /**
00295   * @brief  Stops audio playing and Power do
wn the Audio Codec.
00296   * @param  Option could be one of the follo
wing parameters
00297   *           - CODEC_PDWN_SW: for software
power off (by writing registers).
00298   *                               Then no need
```

```
to reconfigure the Codec after power on.
00299   *            - CODEC_PDWN_HW: completely sh
ut down the codec (physically).
00300   *                                Then need to
reconfigure the Codec after power on.
00301   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00302   */
00303 uint8_t BSP_AUDIO_OUT_Stop(uint32_t Option)
00304 {
00305   /* Call DMA Stop to disable DMA stream bef
ore stopping codec */
00306   HAL_I2S_DMAStop(&hAudioOutI2s);
00307
00308   /* Call Audio Codec Stop function */
00309   if(pAudioDrv->Stop(AUDIO_I2C_ADDRESS, Opti
on) != 0)
00310   {
00311     return AUDIO_ERROR;
00312   }
00313   else
00314   {
00315     if(Option == CODEC_PDWN_HW)
00316     {
00317       /* Wait at least 100us */
00318       HAL_Delay(1);
00319     }
00320     /* Return AUDIO_OK when all operations a
re correctly done */
00321     return AUDIO_OK;
00322   }
00323 }
00324
00325 /**
00326   * @brief  Controls the current audio volum
e level.
00327   * @param  Volume Volume level to be set in
```

```
 percentage from 0% to 100% (0 for
00328   *         Mute and 100 for Max volume leve
l).
00329   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00330   */
00331 uint8_t BSP_AUDIO_OUT_SetVolume(uint8_t Volu
me)
00332 {
00333   /* Call the codec volume control function
with converted volume value */
00334   if(pAudioDrv->SetVolume(AUDIO_I2C_ADDRESS,
 Volume) != 0)
00335   {
00336     return AUDIO_ERROR;
00337   }
00338   else
00339   {
00340     /* Return AUDIO_OK when all operations a
re correctly done */
00341     return AUDIO_OK;
00342   }
00343 }
00344
00345 /**
00346   * @brief  Enables or disables the MUTE mod
e by software
00347   * @param  Cmd could be AUDIO_MUTE_ON to mu
te sound or AUDIO_MUTE_OFF to
00348   *         unmute the codec and restore pre
vious volume level.
00349   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00350   */
00351 uint8_t BSP_AUDIO_OUT_SetMute(uint32_t Cmd)
00352 {
00353   /* Call the Codec Mute function */
```

```
00354   if(pAudioDrv->SetMute(AUDIO_I2C_ADDRESS, C
md) != 0)
00355   {
00356     return AUDIO_ERROR;
00357   }
00358   else
00359   {
00360     /* Return AUDIO_OK when all operations a
re correctly done */
00361     return AUDIO_OK;
00362   }
00363 }
00364
00365 /**
00366   * @brief  Switch dynamically (while audio
file is played) the output target
00367   *         (speaker or headphone).
00368   * @note   This function modifies a global
variable of the audio codec driver: OutputDev.
00369   * @param  Output specifies the audio outpu
t target: OUTPUT_DEVICE_SPEAKER,
00370   *         OUTPUT_DEVICE_HEADPHONE, OUTPUT_
DEVICE_BOTH or OUTPUT_DEVICE_AUTO
00371   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00372   */
00373 uint8_t BSP_AUDIO_OUT_SetOutputMode(uint8_t
Output)
00374 {
00375   /* Call the Codec output Device function */

00376   if(pAudioDrv->SetOutputMode(AUDIO_I2C_ADDR
ESS, Output) != 0)
00377 {
00378     return AUDIO_ERROR;
00379   }
00380   else
```

```
00381   {
00382     /* Return AUDIO_OK when all operations a
re correctly done */
00383     return AUDIO_OK;
00384   }
00385 }
00386
00387 /**
00388   * @brief  Update the audio frequency.
00389   * @param  AudioFreq Audio frequency used t
o play the audio stream.
00390   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00391   */
00392 uint8_t BSP_AUDIO_OUT_SetFrequency(uint32_t
AudioFreq)
00393 {
00394   /* Update the I2S audio frequency configur
ation */
00395   return (I2Sx_Init(AudioFreq));
00396 }
00397
00398 /**
00399   * @brief Tx Transfer completed callbacks
00400   * @param hi2s I2S handle
00401   * @retval None
00402   */
00403 void HAL_I2S_TxCpltCallback(I2S_HandleTypeDe
f *hi2s)
00404 {
00405   if(hi2s->Instance == I2Sx)
00406   {
00407     /* Call the user function which will man
age directly transfer complete*/
00408     BSP_AUDIO_OUT_TransferComplete_CallBack(
);
00409   }
```

```
00410 }
00411
00412 /**
00413   * @brief Tx Transfer Half completed callba
cks
00414   * @param hi2s I2S handle
00415   * @retval None
00416   */
00417 void HAL_I2S_TxHalfCpltCallback(I2S_HandleTy
peDef *hi2s)
00418 {
00419   if(hi2s->Instance == I2Sx)
00420   {
00421     /* Manage the remaining file size and ne
w address offset: This function
00422     should be coded by user (its prototype i
s already declared in stm32303c_eval_audio.h) */
00423     BSP_AUDIO_OUT_HalfTransfer_CallBack();
00424   }
00425 }
00426
00427 /**
00428   * @brief I2S error callbacks
00429   * @param hi2s I2S handle
00430   * @retval None
00431   */
00432 void HAL_I2S_ErrorCallback(I2S_HandleTypeDef
 *hi2s)
00433 {
00434   /* Manage the error generated on DMA: This
 function
00435      should be coded by user (its prototype
is already declared in stm32303c_eval_audio.h) */

00436   if(hi2s->Instance == I2Sx)
00437   {
00438     BSP_AUDIO_OUT_Error_CallBack();
```

```
00439   }
00440 }
00441
00442 /**
00443   * @brief  Manages the DMA full Transfer co
mplete event.
00444   * @retval None
00445   */
00446 __weak void BSP_AUDIO_OUT_TransferComplete_C
allBack(void)
00447 {
00448 }
00449
00450 /**
00451   * @brief  Manages the DMA Half Transfer co
mplete event.
00452   * @retval None
00453   */
00454 __weak void BSP_AUDIO_OUT_HalfTransfer_CallB
ack(void)
00455 {
00456 }
00457
00458 /**
00459   * @brief  Audio OUT Error callback function

00460   * @retval None
00461   */
00462 __weak void BSP_AUDIO_OUT_Error_CallBack(void
)
00463 {
00464 }
00465
00466 /**
00467   * @}
00468   */
00469
```

```
/** @addtogroup STM32303C_EVAL_AUDIO_Private_Functions
  * @{
  */

/*****************************************************************************
                                Static Function
*****************************************************************************/

/**
  * @brief AUDIO OUT I2S MSP Init
  * @retval None
  */
static void I2Sx_MspInit(void)
{
  static DMA_HandleTypeDef hdma_i2sTx;
  GPIO_InitTypeDef  GPIO_InitStruct;
  I2S_HandleTypeDef *hi2s = &hAudioOutI2s;

  /* Enable I2S GPIO clocks */
  I2Sx_MCK_WS_GPIO_CLK_ENABLE();
  I2Sx_SCK_DIN_GPIO_CLK_ENABLE();

  /* I2S pins configuration: SCK and DIN pins ----------------------------*/
  GPIO_InitStruct.Pin         = (I2Sx_SCK_PIN | I2Sx_DIN_PIN);
  GPIO_InitStruct.Mode        = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull        = GPIO_NOPULL;
  GPIO_InitStruct.Speed       = GPIO_SPEED_FREQ_HIGH;
  GPIO_InitStruct.Alternate   = I2Sx_SCK_DIN_AF;
  HAL_GPIO_Init(I2Sx_SCK_DIN_GPIO_PORT, &GPI
```

```
O_InitStruct);
00499
00500   /* I2S pins configuration: WS pin --------
----------------------*/
00501   GPIO_InitStruct.Pin         = I2Sx_WS_PIN;

00502   GPIO_InitStruct.Mode        = GPIO_MODE_AF
_PP;
00503   GPIO_InitStruct.Pull        = GPIO_NOPULL;
00504   GPIO_InitStruct.Speed       = GPIO_SPEED_F
REQ_HIGH;
00505   GPIO_InitStruct.Alternate   = I2Sx_WS_AF;
00506   HAL_GPIO_Init(I2Sx_WS_GPIO_PORT, &GPIO_Ini
tStruct);
00507
00508   /* I2S pins configuration: MCK pin -------
----------------------*/
00509   GPIO_InitStruct.Pin         = I2Sx_MCK_PIN
;
00510   GPIO_InitStruct.Mode        = GPIO_MODE_AF
_PP;
00511   GPIO_InitStruct.Pull        = GPIO_NOPULL;
00512   GPIO_InitStruct.Speed       = GPIO_SPEED_F
REQ_HIGH;
00513   GPIO_InitStruct.Alternate   = I2Sx_MCK_AF;
00514   HAL_GPIO_Init(I2Sx_MCK_GPIO_PORT, &GPIO_In
itStruct);
00515
00516   /* Enable I2S clock */
00517   I2Sx_CLK_ENABLE();
00518
00519   /* Force the I2S peripheral clock reset */
00520   I2Sx_FORCE_RESET();
00521
00522   /* Release the I2S peripheral clock reset
*/
00523   I2Sx_RELEASE_RESET();
```

```
00524
00525   /* Enable the I2S DMA clock */
00526   I2Sx_DMAx_CLK_ENABLE();
00527
00528   /* Configure the hdma_i2sTx handle paramet
ers */
00529   hdma_i2sTx.Init.Direction           = DMA_
MEMORY_TO_PERIPH;
00530   hdma_i2sTx.Init.PeriphInc           = DMA_
PINC_DISABLE;
00531   hdma_i2sTx.Init.MemInc              = DMA_
MINC_ENABLE;
00532   hdma_i2sTx.Init.PeriphDataAlignment = I2Sx
_DMAx_PERIPH_DATA_SIZE;
00533   hdma_i2sTx.Init.MemDataAlignment    = I2Sx
_DMAx_MEM_DATA_SIZE;
00534   hdma_i2sTx.Init.Mode                = DMA_
NORMAL;
00535   hdma_i2sTx.Init.Priority            = DMA_
PRIORITY_HIGH;
00536
00537   hdma_i2sTx.Instance                 = I2Sx
_DMAx_CHANNEL;
00538
00539   /* Associate the DMA handle */
00540   __HAL_LINKDMA(hi2s, hdmatx, hdma_i2sTx);
00541
00542   /* Configure the DMA Stream */
00543   HAL_DMA_Init(&hdma_i2sTx);
00544
00545   /* I2S DMA IRQ Channel configuration */
00546   HAL_NVIC_SetPriority((IRQn_Type)I2Sx_DMAx_
IRQ, AUDIO_OUT_IRQ_PREPRIO, AUDIO_OUT_IRQ_SUBPRIO)
;
00547   HAL_NVIC_EnableIRQ((IRQn_Type)I2Sx_DMAx_IR
Q);
00548 }
```

```
00549
00550 /**
00551   * @brief  Initializes the Audio Codec audi
o interface (I2S)
00552   * @param  AudioFreq Audio frequency to be
configured for the I2S peripheral.
00553   * @retval AUDIO_StatusTypeDef AUDIO Status
00554   */
00555 static AUDIO_StatusTypeDef I2Sx_Init(uint32_
t AudioFreq)
00556 {
00557   /* I2S peripheral configuration */
00558   hAudioOutI2s.Init.AudioFreq   = AudioFreq;
00559   hAudioOutI2s.Init.ClockSource = I2S_CLOCK_
SYSCLK;
00560   hAudioOutI2s.Init.CPOL        = I2S_CPOL_L
OW;
00561   hAudioOutI2s.Init.DataFormat  = I2S_DATAFO
RMAT_16B;
00562   hAudioOutI2s.Init.MCLKOutput  = I2S_MCLKOU
TPUT_ENABLE;
00563   hAudioOutI2s.Init.Mode        = I2S_MODE_M
ASTER_TX;
00564   hAudioOutI2s.Init.Standard    = I2S_STANDA
RD;
00565   hAudioOutI2s.Instance         = I2Sx;
00566
00567   /* Disable I2S block */
00568   __HAL_I2S_DISABLE(&hAudioOutI2s);
00569
00570   /* Initialize the I2S peripheral with the
structure above */
00571   if(HAL_I2S_GetState(&hAudioOutI2s) == HAL_
I2S_STATE_RESET)
00572   {
00573     I2Sx_MspInit();
00574   }
```

```
00575
00576   if (HAL_I2S_Init(&hAudioOutI2s) != HAL_OK)
00577   {
00578     return AUDIO_ERROR;
00579   }
00580
00581   return AUDIO_OK;
00582 }
00583 /**
00584   * @}
00585   */
00586
00587 /**
00588   * @}
00589   */
00590
00591 /**
00592   * @}
00593   */
00594
00595 /**
00596   * @}
00597   */
00598
00599 /**
00600   * @}
00601   */
00602
00603 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32303C_EVAL BSP User Manual

stm32303c_eval_tsensor.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32303c_eval_tsensor.h
00004   * @author  MCD Application Team
00005   * @brief   This file contains all the functions prototypes for the
00006   *          stm32303c_eval_tsensor.c firmware driver.
00007   ******************************************************************************
00008   * @attention
00009   *
00010   * <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>
00011   *
00012   * Redistribution and use in source and binary forms, with or without modification,
00013   * are permitted provided that the following conditions are met:
00014   *   1. Redistributions of source code must retain the above copyright notice,
```

```
00015   *      this list of conditions and the fol
lowing disclaimer.
00016   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00017   *      this list of conditions and the fol
lowing disclaimer in the documentation
00018   *      and/or other materials provided wit
h the distribution.
00019   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00020   *      may be used to endorse or promote p
roducts derived from this software
00021   *      without specific prior written perm
ission.
00022   *
00023   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00024   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00025   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00026   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00027   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00028   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
00029   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00030   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00031   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00032   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00033   *
00034   ****************************************
```

```
**********************************
00035   */
00036
00037 /* Define to prevent recursive inclusion ---
-----------------------------------*/
00038 #ifndef __STM32303C_EVAL_TSENSOR_H
00039 #define __STM32303C_EVAL_TSENSOR_H
00040
00041 #ifdef __cplusplus
00042  extern "C" {
00043 #endif
00044
00045 /* Includes ------------------------------
-----------------------------------*/
00046 #include "stm32303c_eval.h"
00047 #include "../Components/stts751/stts751.h"

00048
00049 /** @addtogroup BSP
00050   * @{
00051   */
00052
00053 /** @addtogroup STM32303C_EVAL
00054   * @{
00055   */
00056
00057 /** @addtogroup STM32303C_EVAL_TSENSOR STM32
303C-EVAL TSENSOR
00058   * @{
00059   */
00060
00061 /** @defgroup STM32303C_EVAL_TSENSOR_Private
_Variables Private Variables
00062   * @{
00063   */
00064 /**
00065   * @}
```

```
00066   */
00067
00068 /** @defgroup STM32303C_EVAL_TSENSOR_Exporte
d_Types Exported Types
00069   * @{
00070   */
00071
00072 /**
00073   * @brief  TSENSOR Status
00074   */
00075 typedef enum
00076 {
00077   TSENSOR_OK = 0,
00078   TSENSOR_ERROR
00079 }TSENSOR_Status_TypDef;
00080
00081 /**
00082   * @}
00083   */
00084
00085 /** @defgroup STM32303C_EVAL_TSENSOR_Exporte
d_Constants Exported Constants
00086   * @{
00087   */
00088 /* Temperature Sensor hardware I2C address */

00089 #define TSENSOR_I2C_ADDRESS     0x90
00090
00091 /* Maximum number of trials use for STTS751_
IsReady function */
00092 #define TSENSOR_MAX_TRIALS      50
00093
00094 /**
00095   * @}
00096   */
00097
00098 /** @defgroup STM32303C_EVAL_I2C_TSENSOR_Exp
```

```
orted_Functions Exported Functions
00099   * @{
00100   */
00101 uint32_t BSP_TSENSOR_Init(void);
00102 uint8_t  BSP_TSENSOR_ReadStatus(void);
00103 uint16_t BSP_TSENSOR_ReadTemp(void);
00104
00105 /**
00106   * @}
00107   */
00108
00109 /** @defgroup STM32303C_EVAL_TSENSOR_Private
_Functions Private Functions
00110   * @{
00111   */
00112 /**
00113   * @}
00114   */
00115
00116 /**
00117   * @}
00118   */
00119
00120 /**
00121   * @}
00122   */
00123
00124 /**
00125   * @}
00126   */
00127
00128 #ifdef __cplusplus
00129 }
00130 #endif
00131
00132 #endif /* __STM32303C_EVAL_TSENSOR_H */
00133
```

```
00134 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM32303C_EVAL BSP User Manual

stm32303c_eval_tsensor.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32303c_eval_tsensor.c
00004   * @author  MCD Application Team
00005   * @brief   This file provides a set of functions needed to manage the I2C TS751
00006   *          temperature sensor mounted on STM32303C-EVAL board .
00007   *          It implements a high level communication layer for read and write
00008   *          from/to this sensor. The needed STM323F30x hardware resources (I2C and
00009   *          GPIO) are defined in stm32303c_eval.h file, and the initialization is
00010   *          performed in TSENSOR_IO_Init() function declared in stm32303c_eval.c
00011   *          file.
00012   *          You can easily tailor this driver to any other development board,
00013   *          by just adapting the defines for hardware resources and
```

```
*          TSENSOR_IO_Init() function.
*
*     +-----------------------------------------------------------------+
*     |                          Pin assignment                         |

*     +-----------------------------------------------------------------+
*     |  STM32F30x I2C Pins                 |   STTS751    |   Pin      |
*     +-----------------------------------------------------------------+
*     | .                                   |   Addr/Therm |    1       |
*     | .                                   |   GND        |    2 (0V)  |
*     | .                                   |   VDD        |    3 (3.3V)|
*     | TSENSOR_I2C_SCL_PIN/ SCL            |   SCL        |    4       |
*     | TSENSOR_I2C_SMBUSALERT_PIN/ SMBUS ALERT|   SMBUS ALERT|    5       |
*     | TSENSOR_I2C_SDA_PIN/ SDA            |   SDA        |    6       |
*     +-----------------------------------------------------------------+
*****************************************************************************
* @attention
*
* <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>
*
* Redistribution and use in source and binary forms, with or without modification,
* are permitted provided that the followin
```

```
g conditions are met:
00035   *   1. Redistributions of source code must
 retain the above copyright notice,
00036   *      this list of conditions and the fol
lowing disclaimer.
00037   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00038   *      this list of conditions and the fol
lowing disclaimer in the documentation
00039   *      and/or other materials provided wit
h the distribution.
00040   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00041   *      may be used to endorse or promote p
roducts derived from this software
00042   *      without specific prior written perm
ission.
00043   *
00044   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00045   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00046   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00047   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00048   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00049   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
00050   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00051   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00052   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00053   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
```

```
 POSSIBILITY OF SUCH DAMAGE.
00054   *
00055   ******************************************
*********************************
00056   */
00057
00058 /* Includes ------------------------------
-----------------------------------*/
00059 #include "stm32303c_eval_tsensor.h"
00060
00061 /** @addtogroup BSP
00062   * @{
00063   */
00064
00065 /** @addtogroup STM32303C_EVAL
00066   * @{
00067   */
00068
00069 /** @addtogroup STM32303C_EVAL_TSENSOR
00070   * @brief      This file includes the TS751
 Temperature Sensor driver of
00071   *             STM32303C-EVAL boards.
00072   * @{
00073   */
00074
00075 /** @addtogroup STM32303C_EVAL_TSENSOR_Priva
te_Variables
00076   * @{
00077   */
00078 static TSENSOR_DrvTypeDef  *tsensor_drv;
00079 /**
00080   * @}
00081   */
00082
00083 /** @addtogroup STM32303C_EVAL_TSENSOR_Priva
te_Functions
00084   * @{
```

```
00085   */
00086
00087 /**
00088   * @brief  Initializes peripherals used by
the I2C Temperature Sensor driver.
00089   * @retval TSENSOR status
00090   */
00091 uint32_t BSP_TSENSOR_Init(void)
00092 {
00093   uint8_t ret = TSENSOR_ERROR;
00094   TSENSOR_InitTypeDef STTS751_InitStructure;
00095
00096   /* Temperature Sensor Initialization */
00097   if(Stts751Drv.IsReady(TSENSOR_I2C_ADDRESS,
TSENSOR_MAX_TRIALS) == HAL_OK)
00098   {
00099     /* Initialize the temperature sensor dri
ver structure */
00100     tsensor_drv = &Stts751Drv;
00101
00102     /* Configure Temperature Sensor : Conver
sion 12 bits in continuous mode at one conversion
per second */
00103     /* Alert outside range Limit Temperature
 12 <-> 24c */
00104     STTS751_InitStructure.AlertMode
    = STTS751_ALERT_ENABLE;
00105     STTS751_InitStructure.ConversionMode
    = STTS751_CONTINUOUS_MODE;
00106     STTS751_InitStructure.ConversionResoluti
on  = STTS751_CONV_12BITS;
00107     STTS751_InitStructure.ConversionRate
    = STTS751_ONE_PER_SECOND;
00108     STTS751_InitStructure.TemperatureLimitHi
gh  = 24;
00109     STTS751_InitStructure.TemperatureLimitLo
w   = 12;
```

```
00110
00111     /* TSENSOR Init */
00112     tsensor_drv->Init(TSENSOR_I2C_ADDRESS, &
STTS751_InitStructure);
00113
00114     ret = TSENSOR_OK;
00115   }
00116   else
00117   {
00118     ret = TSENSOR_ERROR;
00119   }
00120
00121   return ret;
00122 }
00123
00124 /**
00125   * @brief  Returns the Temperature Sensor s
tatus.
00126   * @retval The Temperature Sensor status.
00127   */
00128 uint8_t BSP_TSENSOR_ReadStatus(void)
00129 {
00130   return (tsensor_drv->ReadStatus(TSENSOR_I2
C_ADDRESS));
00131 }
00132
00133 /**
00134   * @brief  Read Temperature register of TS7
51.
00135   * @retval STTS751 measured temperature val
ue.
00136   */
00137 uint16_t BSP_TSENSOR_ReadTemp(void)
00138 {
00139   return tsensor_drv->ReadTemp(TSENSOR_I2C_A
DDRESS);
00140
```

```
00141 }
00142
00143 /**
00144   * @}
00145   */
00146
00147 /**
00148   * @}
00149   */
00150
00151 /**
00152   * @}
00153   */
00154
00155 /**
00156   * @}
00157   */
00158
00159 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32303C_EVAL BSP User Manual

Modules

BSP

Modules

STM32303C-EVAL

This file provides firmware functions to manage Leds, push-buttons, COM ports, SD card and temperature sensor available on STM32303C-EVAL evaluation board from STMicroelectronics.

STM32303C_EVAL BSP User Manual

STM32303C-EVAL

BSP

This file provides firmware functions to manage Leds, push-buttons, COM ports, SD card and temperature sensor available on STM32303C-EVAL evaluation board from STMicroelectronics. More...

Modules

STM32303C-EVAL Common

STM32303C_EVAL AUDIO

This file includes the low layer audio driver available on STM32303C_EVAL evaluation board(MB1019).

STM32303C-EVAL EEPROM

This file includes the I2C EEPROM and SPI EEPROM driver of STM32303C-EVAL board.

STM32303C-EVAL LCD

STM32303C-EVAL SD

STM32303C-EVAL TSENSOR

This file includes the TS751 Temperature Sensor driver of STM32303C-EVAL boards.

Detailed Description

This file provides firmware functions to manage Leds, push-buttons, COM ports, SD card and temperature sensor available on STM32303C-EVAL evaluation board from STMicroelectronics.

STM32303C_EVAL BSP User Manual

STM32303C-EVAL Common

STM32303C-EVAL

Modules

- Bus Operation functions
- Link Operation functions
- Private Constants
- Private Variables
- Exported Types
- Exported Constants
- Exported Functions

STM32303C_EVAL BSP User Manual

Exported Constants

STM32303C-EVAL Common

Modules

STM32303C-EVAL LED

Define for STM32303C_EVAL board.

STM32303C-EVAL BUTTON

STM32303C-EVAL COM

STM32303C-EVAL COMPONENT

STM32303C_EVAL BSP User Manual

STM32303C-EVAL EEPROM

STM32303C-EVAL

This file includes the I2C EEPROM and SPI EEPROM driver of STM32303C-EVAL board. More...

Modules

- Private Types
- Private Variables
- Exported Types
- Exported Constants
- Exported Functions
- STM32303C_EVAL_EEPROM_Private_Functions

Detailed Description

This file includes the I2C EEPROM and SPI EEPROM driver of STM32303C-EVAL board.

STM32303C_EVAL BSP User Manual

STM32303C-EVAL LCD

STM32303C-EVAL

Modules

- Private Defines
- Private Macros
- Private Variables
- Private Functions
- Exported Types
- Exported Constants
- Exported Functions

STM32303C_EVAL BSP User Manual

STM32303C-EVAL TSENSOR

This file includes the TS751 Temperature Sensor driver of STM32303C-EVAL boards. More...

Modules

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

Detailed Description

This file includes the TS751 Temperature Sensor driver of STM32303C-EVAL boards.
