STM32072B_EVAL BSP User Manual

EEPROM_DrvTypeDef Struct Reference

Exported Types

```
#include <stm32072b_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 **63** of file **stm32072b_eval_eeprom.h**.

Field Documentation

uint32_t(* EEPROM_DrvTypeDef::Init)(void)

Definition at line **65** of file **stm32072b_eval_eeprom.h**.

Referenced by **BSP_EEPROM_Init()**.

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

Definition at line **66** of file **stm32072b_eval_eeprom.h**.

Referenced by **BSP_EEPROM_ReadBuffer()**.

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

Definition at line **67** of file **stm32072b_eval_eeprom.h**.

Referenced by **BSP_EEPROM_WriteBuffer()**.

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

- **stm32072b_eval_eeprom.h**

STM32072B_EVAL BSP User Manual

LCD_DrawPropTypeDef Struct Reference

Exported Types

`#include <stm32072b_eval_lcd.h>`

Data Fields

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

Detailed Description

Definition at line **67** of file **stm32072b_eval_lcd.h**.

Field Documentation

uint32_t LCD_DrawPropTypeDef::BackColor

Definition at line 70 of file stm32072b_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 71 of file stm32072b_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 69 of file stm32072b_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:

- stm32072b_eval_lcd.h

Generated on Wed Jul 5 2017 08:56:11 for STM32072B_EVAL BSP
User Manual by  1.7.6.1

STM32072B_EVAL BSP User Manual

Point Struct Reference

Exported Constants

```
#include <stm32072b_eval_lcd.h>
```

Data Fields

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

Detailed Description

Definition at line **88** of file **stm32072b_eval_lcd.h**.

Field Documentation

int16_t Point::X

Definition at line **90** of file **stm32072b_eval_lcd.h**.

Referenced by **BSP_LCD_DrawPolygon()**.

int16_t Point::Y

Definition at line **91** of file **stm32072b_eval_lcd.h**.

Referenced by **BSP_LCD_DrawPolygon()**.

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

- **stm32072b_eval_lcd.h**

STM32072B_EVAL BSP User Manual

SD_CmdAnswer_typedef Struct Reference

Types Definitions

Data Fields

uint8_t	**r1**
uint8_t	**r2**
uint8_t	**r3**
uint8_t	**r4**
uint8_t	**r5**

Detailed Description

Definition at line **110** of file **stm32072b_eval_sd.c**.

Field Documentation

uint8_t SD_CmdAnswer_typedef::r1

Definition at line 111 of file stm32072b_eval_sd.c.

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

uint8_t SD_CmdAnswer_typedef::r2

Definition at line 112 of file stm32072b_eval_sd.c.

Referenced by BSP_SD_GetStatus(), SD_GoIdleState(), and SD_SendCmd().

uint8_t SD_CmdAnswer_typedef::r3

Definition at line 113 of file stm32072b_eval_sd.c.

Referenced by SD_SendCmd().

uint8_t SD_CmdAnswer_typedef::r4

Definition at line 114 of file stm32072b_eval_sd.c.

Referenced by SD_SendCmd().

uint8_t SD_CmdAnswer_typedef::r5

Definition at line **115** of file **stm32072b_eval_sd.c**.

Referenced by **SD_SendCmd()**.

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

- **stm32072b_eval_sd.c**

STM32072B_EVAL BSP User Manual

struct_v1 Struct Reference

Exported Types

```
#include <stm32072b_eval_sd.h>
```

Data Fields

uint8_t	**Reserved1**:2
uint16_t	**DeviceSize**:12
uint8_t	**MaxRdCurrentVDDMin**:3
uint8_t	**MaxRdCurrentVDDMax**:3
uint8_t	**MaxWrCurrentVDDMin**:3
uint8_t	**MaxWrCurrentVDDMax**:3
uint8_t	**DeviceSizeMul**:3

Detailed Description

Definition at line **75** of file **stm32072b_eval_sd.h**.

Field Documentation

uint16_t struct_v1::DeviceSize

Definition at line 78 of file stm32072b_eval_sd.h.

Referenced by BSP_SD_GetCardInfo(), and SD_GetCSDRegister().

uint8_t struct_v1::DeviceSizeMul

Definition at line 83 of file stm32072b_eval_sd.h.

Referenced by BSP_SD_GetCardInfo(), and SD_GetCSDRegister().

uint8_t struct_v1::MaxRdCurrentVDDMax

Definition at line 80 of file stm32072b_eval_sd.h.

Referenced by SD_GetCSDRegister().

uint8_t struct_v1::MaxRdCurrentVDDMin

Definition at line 79 of file stm32072b_eval_sd.h.

Referenced by SD_GetCSDRegister().

uint8_t struct_v1::MaxWrCurrentVDDMax

Definition at line 82 of file stm32072b_eval_sd.h.

Referenced by SD_GetCSDRegister().

uint8_t struct_v1::MaxWrCurrentVDDMin

Definition at line 81 of file stm32072b_eval_sd.h.

Referenced by SD_GetCSDRegister().

uint8_t struct_v1::Reserved1

Definition at line 77 of file stm32072b_eval_sd.h.

Referenced by SD_GetCSDRegister().

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

- stm32072b_eval_sd.h

STM32072B_EVAL BSP User Manual

struct_v2 Struct Reference

Exported Types

```
#include <stm32072b_eval_sd.h>
```

Data Fields

uint8_t	**Reserved1**:6
uint32_t	**DeviceSize**:22
uint8_t	**Reserved2**:1

Detailed Description

Definition at line **87** of file **stm32072b_eval_sd.h**.

Field Documentation

uint32_t struct_v2::DeviceSize

Definition at line 90 of file stm32072b_eval_sd.h.

Referenced by BSP_SD_GetCardInfo(), and SD_GetCSDRegister().

uint8_t struct_v2::Reserved1

Definition at line 89 of file stm32072b_eval_sd.h.

Referenced by SD_GetCSDRegister().

uint8_t struct_v2::Reserved2

Definition at line 91 of file stm32072b_eval_sd.h.

Referenced by SD_GetCSDRegister().

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

- stm32072b_eval_sd.h

STM32072B_EVAL BSP User Manual

SD_CSD Struct Reference

Exported Types

Card Specific Data: CSD Register. More...

```
#include <stm32072b_eval_sd.h>
```

Data Structures

union **csd_version**

Data Fields

uint8_t	**CSDStruct**:2
uint8_t	**Reserved1**:6
uint8_t	**TAAC**:8
uint8_t	**NSAC**:8
uint8_t	**MaxBusClkFrec**:8
uint16_t	**CardComdClasses**:12
uint8_t	**RdBlockLen**:4
uint8_t	**PartBlockRead**:1
uint8_t	**WrBlockMisalign**:1
uint8_t	**RdBlockMisalign**:1
uint8_t	**DSRImpl**:1
union **SD_CSD::csd_version**	**version**
uint8_t	**EraseSingleBlockEnable**:1
uint8_t	**EraseSectorSize**:7
uint8_t	**WrProtectGrSize**:7
uint8_t	**WrProtectGrEnable**:1
uint8_t	**Reserved2**:2
uint8_t	**WrSpeedFact**:3
uint8_t	**MaxWrBlockLen**:4
uint8_t	**WriteBlockPartial**:1
uint8_t	**Reserved3**:5
uint8_t	**FileFormatGrouop**:1
uint8_t	**CopyFlag**:1
uint8_t	**PermWrProtect**:1
uint8_t	**TempWrProtect**:1
uint8_t	**FileFormat**:2
uint8_t	**Reserved4**:2
uint8_t	**crc**:7
uint8_t	**Reserved5**:1

Detailed Description

Card Specific Data: CSD Register.

Definition at line **97** of file **stm32072b_eval_sd.h**.

Field Documentation

uint16_t SD_CSD::CardComdClasses

Definition at line 105 of file stm32072b_eval_sd.h.

Referenced by SD_GetCSDRegister().

uint8_t SD_CSD::CopyFlag

Definition at line 128 of file stm32072b_eval_sd.h.

Referenced by SD_GetCSDRegister().

uint8_t SD_CSD::crc

Definition at line 133 of file stm32072b_eval_sd.h.

Referenced by SD_GetCSDRegister().

uint8_t SD_CSD::CSDStruct

Definition at line 100 of file stm32072b_eval_sd.h.

Referenced by SD_GetCSDRegister().

uint8_t SD_CSD::DSRImpl

Definition at line 110 of file stm32072b_eval_sd.h.

Referenced by SD_GetCSDRegister().

uint8_t SD_CSD::EraseSectorSize

Definition at line **119** of file **stm32072b_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

uint8_t SD_CSD::EraseSingleBlockEnable

Definition at line **118** of file **stm32072b_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

uint8_t SD_CSD::FileFormat

Definition at line **131** of file **stm32072b_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

uint8_t SD_CSD::FileFormatGrouop

Definition at line **127** of file **stm32072b_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

uint8_t SD_CSD::MaxBusClkFrec

Definition at line **104** of file **stm32072b_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

uint8_t SD_CSD::MaxWrBlockLen

Definition at line **124** of file **stm32072b_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

uint8_t SD_CSD::NSAC

Definition at line **103** of file **stm32072b_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

uint8_t SD_CSD::PartBlockRead

Definition at line **107** of file **stm32072b_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

uint8_t SD_CSD::PermWrProtect

Definition at line **129** of file **stm32072b_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

uint8_t SD_CSD::RdBlockLen

Definition at line **106** of file **stm32072b_eval_sd.h**.

Referenced by **BSP_SD_GetCardInfo()**, and **SD_GetCSDRegister()**.

uint8_t SD_CSD::RdBlockMisalign

Definition at line **109** of file **stm32072b_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

uint8_t SD_CSD::Reserved1

Definition at line **101** of file **stm32072b_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

uint8_t SD_CSD::Reserved2

Definition at line **122** of file **stm32072b_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

uint8_t SD_CSD::Reserved3

Definition at line **126** of file **stm32072b_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

uint8_t SD_CSD::Reserved4

Definition at line **132** of file **stm32072b_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

uint8_t SD_CSD::Reserved5

Definition at line **134** of file **stm32072b_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

uint8_t SD_CSD::TAAC

Definition at line 102 of file stm32072b_eval_sd.h.

Referenced by SD_GetCSDRegister().

uint8_t SD_CSD::TempWrProtect

Definition at line 130 of file stm32072b_eval_sd.h.

Referenced by SD_GetCSDRegister().

union SD_CSD::csd_version SD_CSD::version

Referenced by BSP_SD_GetCardInfo(), and SD_GetCSDRegister().

uint8_t SD_CSD::WrBlockMisalign

Definition at line 108 of file stm32072b_eval_sd.h.

Referenced by SD_GetCSDRegister().

uint8_t SD_CSD::WriteBlockPartial

Definition at line 125 of file stm32072b_eval_sd.h.

Referenced by SD_GetCSDRegister().

uint8_t SD_CSD::WrProtectGrEnable

Definition at line 121 of file stm32072b_eval_sd.h.

Referenced by **SD_GetCSDRegister()**.

uint8_t SD_CSD::WrProtectGrSize

Definition at line **120** of file **stm32072b_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

uint8_t SD_CSD::WrSpeedFact

Definition at line **123** of file **stm32072b_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

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

- **stm32072b_eval_sd.h**

STM32072B_EVAL BSP User Manual

SD_CID Struct Reference

Exported Types

Card Identification Data: CID Register. More...

```
#include <stm32072b_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 **141** of file **stm32072b_eval_sd.h**.

Field Documentation

__IO uint8_t SD_CID::CID_CRC

Definition at line 151 of file stm32072b_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint16_t SD_CID::ManufactDate

Definition at line 150 of file stm32072b_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint8_t SD_CID::ManufacturerID

Definition at line 143 of file stm32072b_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint16_t SD_CID::OEM_AppliID

Definition at line 144 of file stm32072b_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint32_t SD_CID::ProdName1

Definition at line 145 of file stm32072b_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint8_t SD_CID::ProdName2

Definition at line 146 of file stm32072b_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint8_t SD_CID::ProdRev

Definition at line 147 of file stm32072b_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint32_t SD_CID::ProdSN

Definition at line 148 of file stm32072b_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint8_t SD_CID::Reserved1

Definition at line 149 of file stm32072b_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint8_t SD_CID::Reserved2

Definition at line 152 of file stm32072b_eval_sd.h.

Referenced by SD_GetCIDRegister().

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

- **stm32072b_eval_sd.h**

STM32072B_EVAL BSP User Manual

SD_CardInfo Struct Reference

Exported Types

SD Card information. More...

```
#include <stm32072b_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 **158** of file **stm32072b_eval_sd.h**.

Field Documentation

uint32_t SD_CardInfo::CardBlockSize

Definition at line **163** of file **stm32072b_eval_sd.h**.

Referenced by **BSP_SD_GetCardInfo()**.

uint32_t SD_CardInfo::CardCapacity

Definition at line **162** of file **stm32072b_eval_sd.h**.

Referenced by **BSP_SD_GetCardInfo()**.

SD_CID SD_CardInfo::Cid

Definition at line **161** of file **stm32072b_eval_sd.h**.

Referenced by **BSP_SD_GetCardInfo()**.

SD_CSD SD_CardInfo::Csd

Definition at line **160** of file **stm32072b_eval_sd.h**.

Referenced by **BSP_SD_GetCardInfo()**.

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

- **stm32072b_eval_sd.h**

STM32072B_EVAL BSP User Manual

Exported Macro

STM32072B_EVAL SD

Generated on Wed Jul 5 2017 08:56:11 for STM32072B_EVAL BSP User Manual by doxygen 1.7.6.1

STM32072B_EVAL BSP User Manual

LINK Operations Functions

STM32072B_EVAL SD

Generated on Wed Jul 5 2017 08:56:11 for STM32072B_EVAL BSP User Manual by doxygen 1.7.6.1

STM32072B_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**
- CopyFlag : **SD_CSD**
- crc : **SD_CSD**
- Csd : **SD_CardInfo**
- CSDStruct : **SD_CSD**

- d -

- DeviceSize : **struct_v1** , **struct_v2**
- DeviceSizeMul : **struct_v1**
- DSRImpl : **SD_CSD**

- e -

- EraseSectorSize : **SD_CSD**
- EraseSingleBlockEnable : **SD_CSD**

- f -

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

- i -

- Init : **EEPROM_DrvTypeDef**

- m -

- ManufactDate : **SD_CID**
- ManufacturerID : **SD_CID**
- MaxBusClkFrec : **SD_CSD**
- MaxRdCurrentVDDMax : **struct_v1**
- MaxRdCurrentVDDMin : **struct_v1**
- MaxWrBlockLen : **SD_CSD**
- MaxWrCurrentVDDMax : **struct_v1**
- MaxWrCurrentVDDMin : **struct_v1**

- 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 -

- r1 : **SD_CmdAnswer_typedef**
- r2 : **SD_CmdAnswer_typedef**
- r3 : **SD_CmdAnswer_typedef**
- r4 : **SD_CmdAnswer_typedef**
- r5 : **SD_CmdAnswer_typedef**
- RdBlockLen : **SD_CSD**
- RdBlockMisalign : **SD_CSD**
- ReadBuffer : **EEPROM_DrvTypeDef**
- Reserved1 : **SD_CSD** , **struct_v1** , **struct_v2** , **SD_CID**
- Reserved2 : **SD_CSD** , **SD_CID** , **struct_v2**
- Reserved3 : **SD_CSD**
- Reserved4 : **SD_CSD**
- Reserved5 : **SD_CSD**

- t -

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

- v -

- v1 : **SD_CSD::csd_version**
- v2 : **SD_CSD::csd_version**
- version : **SD_CSD**

- w -

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

- x -

- X : **Point**

- y -

- Y : **Point**

STM32072B_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**
- CopyFlag : **SD_CSD**
- crc : **SD_CSD**
- Csd : **SD_CardInfo**
- CSDStruct : **SD_CSD**

- d -

- DeviceSize : **struct_v1** , **struct_v2**
- DeviceSizeMul : **struct_v1**
- DSRImpl : **SD_CSD**

- e -

- EraseSectorSize : **SD_CSD**
- EraseSingleBlockEnable : **SD_CSD**

- f -

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

- i -

- Init : **EEPROM_DrvTypeDef**

- m -

- ManufactDate : **SD_CID**
- ManufacturerID : **SD_CID**
- MaxBusClkFrec : **SD_CSD**
- MaxRdCurrentVDDMax : **struct_v1**
- MaxRdCurrentVDDMin : **struct_v1**
- MaxWrBlockLen : **SD_CSD**
- MaxWrCurrentVDDMax : **struct_v1**
- MaxWrCurrentVDDMin : **struct_v1**

- 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 -

- r1 : **SD_CmdAnswer_typedef**
- r2 : **SD_CmdAnswer_typedef**
- r3 : **SD_CmdAnswer_typedef**
- r4 : **SD_CmdAnswer_typedef**
- r5 : **SD_CmdAnswer_typedef**
- RdBlockLen : **SD_CSD**
- RdBlockMisalign : **SD_CSD**
- ReadBuffer : **EEPROM_DrvTypeDef**
- Reserved1 : **SD_CSD** , **struct_v1** , **struct_v2** , **SD_CID**
- Reserved2 : **SD_CSD** , **SD_CID** , **struct_v2**
- Reserved3 : **SD_CSD**
- Reserved4 : **SD_CSD**
- Reserved5 : **SD_CSD**

- t -

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

- v -

- v1 : **SD_CSD::csd_version**
- v2 : **SD_CSD::csd_version**
- version : **SD_CSD**

- w -

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

- x -

- X : **Point**

- y -

- Y : **Point**

STM32072B_EVAL BSP User Manual

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

- _ -

- __STM32072B_EVAL_BSP_VERSION : **stm32072b_eval.c**
- __STM32072B_EVAL_BSP_VERSION_MAIN : **stm32072b_eval.c**
- __STM32072B_EVAL_BSP_VERSION_RC : **stm32072b_eval.c**
- __STM32072B_EVAL_BSP_VERSION_SUB1 : **stm32072b_eval.c**
- __STM32072B_EVAL_BSP_VERSION_SUB2 : **stm32072b_eval.c**

STM32072B_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 : **stm32072b_eval_lcd.c**

STM32072B_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 : **stm32072b_eval_lcd.c**
- BSP_COM_Init() : **stm32072b_eval.c**
- BSP_EEPROM_Init() : **stm32072b_eval_eeprom.c**
- BSP_EEPROM_M24LR64 : **stm32072b_eval_eeprom.h**
- BSP_EEPROM_ReadBuffer() : **stm32072b_eval_eeprom.c**
- BSP_EEPROM_TIMEOUT_UserCallback() : **stm32072b_eval_eeprom.c**
- BSP_EEPROM_WriteBuffer() : **stm32072b_eval_eeprom.c**
- BSP_GetVersion() : **stm32072b_eval.c**
- BSP_JOY_GetState() : **stm32072b_eval.c**
- BSP_JOY_Init() : **stm32072b_eval.c**
- BSP_LCD_Clear() : **stm32072b_eval_lcd.c**
- BSP_LCD_ClearStringLine() : **stm32072b_eval_lcd.c**
- BSP_LCD_DisplayChar() : **stm32072b_eval_lcd.c**
- BSP_LCD_DisplayOff() : **stm32072b_eval_lcd.c**
- BSP_LCD_DisplayOn() : **stm32072b_eval_lcd.c**
- BSP_LCD_DisplayStringAt() : **stm32072b_eval_lcd.c**
- BSP_LCD_DisplayStringAtLine() : **stm32072b_eval_lcd.c**
- BSP_LCD_DrawBitmap() : **stm32072b_eval_lcd.c**

- BSP_LCD_DrawCircle() : **stm32072b_eval_lcd.c**
- BSP_LCD_DrawEllipse() : **stm32072b_eval_lcd.c**
- BSP_LCD_DrawHLine() : **stm32072b_eval_lcd.c**
- BSP_LCD_DrawLine() : **stm32072b_eval_lcd.c**
- BSP_LCD_DrawPolygon() : **stm32072b_eval_lcd.c**
- BSP_LCD_DrawRect() : **stm32072b_eval_lcd.c**
- BSP_LCD_DrawVLine() : **stm32072b_eval_lcd.c**
- BSP_LCD_FillCircle() : **stm32072b_eval_lcd.c**
- BSP_LCD_FillEllipse() : **stm32072b_eval_lcd.c**
- BSP_LCD_FillRect() : **stm32072b_eval_lcd.c**
- BSP_LCD_GetBackColor() : **stm32072b_eval_lcd.c**
- BSP_LCD_GetFont() : **stm32072b_eval_lcd.c**
- BSP_LCD_GetTextColor() : **stm32072b_eval_lcd.c**
- BSP_LCD_GetXSize() : **stm32072b_eval_lcd.c**
- BSP_LCD_GetYSize() : **stm32072b_eval_lcd.c**
- BSP_LCD_Init() : **stm32072b_eval_lcd.c**
- BSP_LCD_ReadPixel() : **stm32072b_eval_lcd.c**
- BSP_LCD_SetBackColor() : **stm32072b_eval_lcd.c** , **stm32072b_eval_lcd.h**
- BSP_LCD_SetFont() : **stm32072b_eval_lcd.c**
- BSP_LCD_SetTextColor() : **stm32072b_eval_lcd.h** , **stm32072b_eval_lcd.c**
- BSP_LED_Init() : **stm32072b_eval.c**
- BSP_LED_Off() : **stm32072b_eval.c**
- BSP_LED_On() : **stm32072b_eval.c**
- BSP_LED_Toggle() : **stm32072b_eval.c**
- BSP_PB_GetState() : **stm32072b_eval.c**
- BSP_PB_Init() : **stm32072b_eval.c**
- BSP_SD_Erase() : **stm32072b_eval_sd.c**
- BSP_SD_ERROR : **stm32072b_eval_sd.h**
- BSP_SD_GetCardInfo() : **stm32072b_eval_sd.c**
- BSP_SD_GetStatus() : **stm32072b_eval_sd.c**
- BSP_SD_Init() : **stm32072b_eval_sd.c**
- BSP_SD_IsDetected() : **stm32072b_eval_sd.c**
- BSP_SD_OK : **stm32072b_eval_sd.h**
- BSP_SD_ReadBlocks() : **stm32072b_eval_sd.c**
- BSP_SD_TIMEOUT : **stm32072b_eval_sd.h**

- BSP_SD_WriteBlocks() : **stm32072b_eval_sd.c**
- BSP_TSENSOR_Init() : **stm32072b_eval_tsensor.c**
- BSP_TSENSOR_ReadStatus() : **stm32072b_eval_tsensor.c**
- BSP_TSENSOR_ReadTemp() : **stm32072b_eval_tsensor.c**
- BUTTON_IRQn : **stm32072b_eval.c**
- BUTTON_MODE_EXTI : **stm32072b_eval.h**
- BUTTON_MODE_GPIO : **stm32072b_eval.h**
- BUTTON_PIN : **stm32072b_eval.c**
- BUTTON_PORT : **stm32072b_eval.c**
- BUTTON_TAMPER : **stm32072b_eval.h**
- Button_TypeDef : **stm32072b_eval.h**
- ButtonMode_TypeDef : **stm32072b_eval.h**
- BUTTONn : **stm32072b_eval.h**

STM32072B_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 : **stm32072b_eval_lcd.h**
- COM1 : **stm32072b_eval.h**
- COM_RX_AF : **stm32072b_eval.c**
- COM_RX_PIN : **stm32072b_eval.c**
- COM_RX_PORT : **stm32072b_eval.c**
- COM_TX_AF : **stm32072b_eval.c**
- COM_TX_PIN : **stm32072b_eval.c**
- COM_TX_PORT : **stm32072b_eval.c**
- COM_TypeDef : **stm32072b_eval.h**
- COM_USART : **stm32072b_eval.c**
- COMn : **stm32072b_eval.h**
- COMx_CLK_DISABLE : **stm32072b_eval.h**
- COMx_CLK_ENABLE : **stm32072b_eval.h**
- COMx_CTS_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- COMx_CTS_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- COMx_RTS_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- COMx_RTS_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- COMx_RX_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- COMx_RX_GPIO_CLK_ENABLE : **stm32072b_eval.h**

- COMx_TX_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- COMx_TX_GPIO_CLK_ENABLE : **stm32072b_eval.h**

STM32072B_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 -

- DOWN_JOY_EXTI_IRQn : **stm32072b_eval.h**
- DOWN_JOY_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- DOWN_JOY_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- DOWN_JOY_GPIO_PORT : **stm32072b_eval.h**
- DOWN_JOY_PIN : **stm32072b_eval.h**
- DrawProp : **stm32072b_eval_lcd.c**

STM32072B_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 : **stm32072b_eval_eeprom.h**
- EEPROM_ADDRESS_M24LR64_A02 : **stm32072b_eval_eeprom.h**
- EEPROM_FAIL : **stm32072b_eval_eeprom.h**
- EEPROM_I2C_Drv : **stm32072b_eval_eeprom.c**
- EEPROM_I2C_Init() : **stm32072b_eval_eeprom.c**
- EEPROM_I2C_ReadBuffer() : **stm32072b_eval_eeprom.c**
- EEPROM_I2C_WaitEepromStandbyState() : **stm32072b_eval_eeprom.c**
- EEPROM_I2C_WritePage() : **stm32072b_eval_eeprom.c**
- EEPROM_IO_Init() : **stm32072b_eval.c** , **stm32072b_eval_eeprom.h**
- EEPROM_IO_IsDeviceReady() : **stm32072b_eval.c** , **stm32072b_eval_eeprom.h**
- EEPROM_IO_ReadData() : **stm32072b_eval.c** , **stm32072b_eval_eeprom.h**
- EEPROM_IO_WriteData() : **stm32072b_eval_eeprom.h** , **stm32072b_eval.c**

- EEPROM_MAX_TRIALS : **stm32072b_eval_eeprom.h**
- EEPROM_OK : **stm32072b_eval_eeprom.h**
- EEPROM_PAGESIZE_M24LR64 : **stm32072b_eval_eeprom.h**
- EEPROM_SelectedDevice : **stm32072b_eval_eeprom.c**
- EEPROM_TIMEOUT : **stm32072b_eval_eeprom.h**
- EEPROMAddress : **stm32072b_eval_eeprom.c**
- EEPROMDataRead : **stm32072b_eval_eeprom.c**
- EEPROMDataWrite : **stm32072b_eval_eeprom.c**
- EEPROMPageSize : **stm32072b_eval_eeprom.c**
- EVAL_COM1 : **stm32072b_eval.h**
- EVAL_COM1_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_COM1_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_COM1_CTS_AF : **stm32072b_eval.h**
- EVAL_COM1_CTS_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_COM1_CTS_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_COM1_CTS_GPIO_PORT : **stm32072b_eval.h**
- EVAL_COM1_CTS_PIN : **stm32072b_eval.h**
- EVAL_COM1_IRQn : **stm32072b_eval.h**
- EVAL_COM1_RTS_AF : **stm32072b_eval.h**
- EVAL_COM1_RTS_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_COM1_RTS_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_COM1_RTS_GPIO_PORT : **stm32072b_eval.h**
- EVAL_COM1_RTS_PIN : **stm32072b_eval.h**
- EVAL_COM1_RX_AF : **stm32072b_eval.h**
- EVAL_COM1_RX_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_COM1_RX_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_COM1_RX_GPIO_PORT : **stm32072b_eval.h**
- EVAL_COM1_RX_PIN : **stm32072b_eval.h**
- EVAL_COM1_TX_AF : **stm32072b_eval.h**
- EVAL_COM1_TX_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_COM1_TX_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_COM1_TX_GPIO_PORT : **stm32072b_eval.h**
- EVAL_COM1_TX_PIN : **stm32072b_eval.h**
- EVAL_I2C1 : **stm32072b_eval.h**
- EVAL_I2C1_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_I2C1_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_I2C1_FORCE_RESET : **stm32072b_eval.h**

- EVAL_I2C1_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_I2C1_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_I2C1_GPIO_PORT : **stm32072b_eval.h**
- EVAL_I2C1_RELEASE_RESET : **stm32072b_eval.h**
- EVAL_I2C1_SCL_PIN : **stm32072b_eval.h**
- EVAL_I2C1_SCL_SDA_AF : **stm32072b_eval.h**
- EVAL_I2C1_SDA_PIN : **stm32072b_eval.h**
- EVAL_I2C1_TIMEOUT_MAX : **stm32072b_eval.h**
- EVAL_I2C2 : **stm32072b_eval.h**
- EVAL_I2C2_AF : **stm32072b_eval.h**
- EVAL_I2C2_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_I2C2_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_I2C2_FORCE_RESET : **stm32072b_eval.h**
- EVAL_I2C2_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_I2C2_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_I2C2_GPIO_PORT : **stm32072b_eval.h**
- EVAL_I2C2_IRQn : **stm32072b_eval.h**
- EVAL_I2C2_RELEASE_RESET : **stm32072b_eval.h**
- EVAL_I2C2_SCL_PIN : **stm32072b_eval.h**
- EVAL_I2C2_SDA_PIN : **stm32072b_eval.h**
- EVAL_I2C2_TIMEOUT_MAX : **stm32072b_eval.h**
- EVAL_SPIx : **stm32072b_eval.h**
- EVAL_SPIx_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_SPIx_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_SPIx_FORCE_RESET : **stm32072b_eval.h**
- EVAL_SPIx_MISO_AF : **stm32072b_eval.h**
- EVAL_SPIx_MISO_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_SPIx_MISO_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_SPIx_MISO_GPIO_PORT : **stm32072b_eval.h**
- EVAL_SPIx_MISO_PIN : **stm32072b_eval.h**
- EVAL_SPIx_MOSI_AF : **stm32072b_eval.h**
- EVAL_SPIx_MOSI_DIR_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_SPIx_MOSI_DIR_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_SPIx_MOSI_DIR_GPIO_PORT : **stm32072b_eval.h**
- EVAL_SPIx_MOSI_DIR_PIN : **stm32072b_eval.h**

- EVAL_SPIx_MOSI_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_SPIx_MOSI_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_SPIx_MOSI_GPIO_PORT : **stm32072b_eval.h**
- EVAL_SPIx_MOSI_PIN : **stm32072b_eval.h**
- EVAL_SPIx_RELEASE_RESET : **stm32072b_eval.h**
- EVAL_SPIx_SCK_AF : **stm32072b_eval.h**
- EVAL_SPIx_SCK_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_SPIx_SCK_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_SPIx_SCK_GPIO_PORT : **stm32072b_eval.h**
- EVAL_SPIx_SCK_PIN : **stm32072b_eval.h**
- EVAL_SPIx_TIMEOUT_MAX : **stm32072b_eval.h**

STM32072B_EVAL BSP User Manual

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

- f -

- flag_SDHC : **stm32072b_eval_sd.c**

STM32072B_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 -

- HDMI_CEC_HPD_SINK_CLK_DISABLE : **stm32072b_eval.h**
- HDMI_CEC_HPD_SINK_CLK_ENABLE : **stm32072b_eval.h**
- HDMI_CEC_HPD_SINK_GPIO_PORT : **stm32072b_eval.h**
- HDMI_CEC_HPD_SINK_PIN : **stm32072b_eval.h**
- HDMI_CEC_HPD_SOURCE_CLK_DISABLE : **stm32072b_eval.h**
- HDMI_CEC_HPD_SOURCE_CLK_ENABLE : **stm32072b_eval.h**
- HDMI_CEC_HPD_SOURCE_GPIO_PORT : **stm32072b_eval.h**
- HDMI_CEC_HPD_SOURCE_PIN : **stm32072b_eval.h**
- HDMI_CEC_I2C_ADDRESS : **stm32072b_eval.h**
- HDMI_CEC_IO_Init() : **stm32072b_eval.c**
- HDMI_CEC_IO_ReadData() : **stm32072b_eval.c**
- HDMI_CEC_IO_WriteData() : **stm32072b_eval.c**
- HDMI_CEC_IRQn : **stm32072b_eval.h**
- HDMI_CEC_LINE_AF : **stm32072b_eval.h**
- HDMI_CEC_LINE_CLK_DISABLE : **stm32072b_eval.h**
- HDMI_CEC_LINE_CLK_ENABLE : **stm32072b_eval.h**
- HDMI_CEC_LINE_GPIO_PORT : **stm32072b_eval.h**
- HDMI_CEC_LINE_PIN : **stm32072b_eval.h**
- heval_I2c1 : **stm32072b_eval.c**

- heval_I2c2 : **stm32072b_eval.c**
- heval_Spi : **stm32072b_eval.c**

STM32072B_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 -

- I2C1_Error() : **stm32072b_eval.c**
- I2C1_Init() : **stm32072b_eval.c**
- I2C1_IsDeviceReady() : **stm32072b_eval.c**
- I2C1_MspInit() : **stm32072b_eval.c**
- I2C1_ReadBuffer() : **stm32072b_eval.c**
- I2C1_TIMING : **stm32072b_eval.h**
- I2C1_TransmitData() : **stm32072b_eval.c**
- I2C1_WriteBuffer() : **stm32072b_eval.c**
- I2c1Timeout : **stm32072b_eval.c**
- I2C2_Error() : **stm32072b_eval.c**
- I2C2_Init() : **stm32072b_eval.c**
- I2C2_MspInit() : **stm32072b_eval.c**
- I2C2_ReceiveData() : **stm32072b_eval.c**
- I2C2_TIMING : **stm32072b_eval.h**
- I2c2Timeout : **stm32072b_eval.c**

STM32072B_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 : **stm32072b_eval.h**
- JOY_IRQn : **stm32072b_eval.c**
- JOY_LEFT : **stm32072b_eval.h**
- JOY_MODE_EXTI : **stm32072b_eval.h**
- JOY_MODE_GPIO : **stm32072b_eval.h**
- JOY_NONE : **stm32072b_eval.h**
- JOY_PIN : **stm32072b_eval.c**
- JOY_PORT : **stm32072b_eval.c**
- JOY_RIGHT : **stm32072b_eval.h**
- JOY_SEL : **stm32072b_eval.h**
- JOY_UP : **stm32072b_eval.h**
- JOYMode_TypeDef : **stm32072b_eval.h**
- JOYn : **stm32072b_eval.h**
- JOYState_TypeDef : **stm32072b_eval.h**
- JOYx_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- JOYx_GPIO_CLK_ENABLE : **stm32072b_eval.h**

User Manual by  1.7.6.1

STM32072B_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 : **stm32072b_eval_lcd.h**
- LCD_COLOR_BLUE : **stm32072b_eval_lcd.h**
- LCD_COLOR_BROWN : **stm32072b_eval_lcd.h**
- LCD_COLOR_CYAN : **stm32072b_eval_lcd.h**
- LCD_COLOR_DARKBLUE : **stm32072b_eval_lcd.h**
- LCD_COLOR_DARKCYAN : **stm32072b_eval_lcd.h**
- LCD_COLOR_DARKGRAY : **stm32072b_eval_lcd.h**
- LCD_COLOR_DARKGREEN : **stm32072b_eval_lcd.h**
- LCD_COLOR_DARKMAGENTA : **stm32072b_eval_lcd.h**
- LCD_COLOR_DARKRED : **stm32072b_eval_lcd.h**
- LCD_COLOR_DARKYELLOW : **stm32072b_eval_lcd.h**
- LCD_COLOR_GRAY : **stm32072b_eval_lcd.h**
- LCD_COLOR_GREEN : **stm32072b_eval_lcd.h**
- LCD_COLOR_LIGHTBLUE : **stm32072b_eval_lcd.h**
- LCD_COLOR_LIGHTCYAN : **stm32072b_eval_lcd.h**
- LCD_COLOR_LIGHTGRAY : **stm32072b_eval_lcd.h**
- LCD_COLOR_LIGHTGREEN : **stm32072b_eval_lcd.h**
- LCD_COLOR_LIGHTMAGENTA : **stm32072b_eval_lcd.h**
- LCD_COLOR_LIGHTRED : **stm32072b_eval_lcd.h**

- LCD_COLOR_LIGHTYELLOW : **stm32072b_eval_lcd.h**
- LCD_COLOR_MAGENTA : **stm32072b_eval_lcd.h**
- LCD_COLOR_ORANGE : **stm32072b_eval_lcd.h**
- LCD_COLOR_RED : **stm32072b_eval_lcd.h**
- LCD_COLOR_WHITE : **stm32072b_eval_lcd.h**
- LCD_COLOR_YELLOW : **stm32072b_eval_lcd.h**
- LCD_CS_HIGH : **stm32072b_eval.h**
- LCD_CS_LOW : **stm32072b_eval.h**
- LCD_DEFAULT_FONT : **stm32072b_eval_lcd.h**
- LCD_Delay() : **stm32072b_eval.c**
- LCD_DrawChar() : **stm32072b_eval_lcd.c**
- LCD_DrawPixel() : **stm32072b_eval_lcd.c**
- lcd_drv : **stm32072b_eval_lcd.c**
- LCD_ERROR : **stm32072b_eval_lcd.h**
- LCD_IO_Init() : **stm32072b_eval.c**
- LCD_IO_ReadData() : **stm32072b_eval.c**
- LCD_IO_WriteMultipleData() : **stm32072b_eval.c**
- LCD_IO_WriteReg() : **stm32072b_eval.c**
- LCD_NCS_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- LCD_NCS_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- LCD_NCS_GPIO_PORT : **stm32072b_eval.h**
- LCD_NCS_PIN : **stm32072b_eval.h**
- LCD_OK : **stm32072b_eval_lcd.h**
- LCD_READ_REG : **stm32072b_eval.c**
- LCD_SetDisplayWindow() : **stm32072b_eval_lcd.c**
- LCD_TIMEOUT : **stm32072b_eval_lcd.h**
- LCD_WRITE_REG : **stm32072b_eval.c**
- LED1 : **stm32072b_eval.h**
- LED1_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- LED1_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- LED1_GPIO_PORT : **stm32072b_eval.h**
- LED1_PIN : **stm32072b_eval.h**
- LED2 : **stm32072b_eval.h**
- LED2_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- LED2_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- LED2_GPIO_PORT : **stm32072b_eval.h**
- LED2_PIN : **stm32072b_eval.h**

- LED3 : **stm32072b_eval.h**
- LED3_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- LED3_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- LED3_GPIO_PORT : **stm32072b_eval.h**
- LED3_PIN : **stm32072b_eval.h**
- LED4 : **stm32072b_eval.h**
- LED4_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- LED4_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- LED4_GPIO_PORT : **stm32072b_eval.h**
- LED4_PIN : **stm32072b_eval.h**
- LED_BLUE : **stm32072b_eval.h**
- LED_GREEN : **stm32072b_eval.h**
- LED_ORANGE : **stm32072b_eval.h**
- LED_PIN : **stm32072b_eval.c**
- LED_PORT : **stm32072b_eval.c**
- LED_RED : **stm32072b_eval.h**
- Led_TypeDef : **stm32072b_eval.h**
- LEDn : **stm32072b_eval.h**
- LEDx_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- LEDx_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- LEFT_JOY_EXTI_IRQn : **stm32072b_eval.h**
- LEFT_JOY_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- LEFT_JOY_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- LEFT_JOY_GPIO_PORT : **stm32072b_eval.h**
- LEFT_JOY_PIN : **stm32072b_eval.h**
- LEFT_MODE : **stm32072b_eval_lcd.h**
- Line_ModeTypdef : **stm32072b_eval_lcd.h**

STM32072B_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 : **stm32072b_eval_lcd.c**
- MAX_WIDTH_FONT : **stm32072b_eval_lcd.c**
- MSD_ERROR : **stm32072b_eval_sd.h**
- MSD_OK : **stm32072b_eval_sd.h**

STM32072B_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 : **stm32072b_eval_lcd.c**

STM32072B_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 -

- POLY_X : **stm32072b_eval_lcd.c**
- POLY_Y : **stm32072b_eval_lcd.c**
- pPoint : **stm32072b_eval_lcd.h**

STM32072B_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 : **stm32072b_eval.c**
- RIGHT_JOY_EXTI_IRQn : **stm32072b_eval.h**
- RIGHT_JOY_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- RIGHT_JOY_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- RIGHT_JOY_GPIO_PORT : **stm32072b_eval.h**
- RIGHT_JOY_PIN : **stm32072b_eval.h**
- RIGHT_MODE : **stm32072b_eval_lcd.h**

STM32072B_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_ANSWER_R1_EXPECTED : **stm32072b_eval_sd.c**
- SD_ANSWER_R1B_EXPECTED : **stm32072b_eval_sd.c**
- SD_ANSWER_R2_EXPECTED : **stm32072b_eval_sd.c**
- SD_ANSWER_R3_EXPECTED : **stm32072b_eval_sd.c**
- SD_ANSWER_R4R5_EXPECTED : **stm32072b_eval_sd.c**
- SD_ANSWER_R7_EXPECTED : **stm32072b_eval_sd.c**
- SD_Answer_type : **stm32072b_eval_sd.c**
- SD_BLOCK_SIZE : **stm32072b_eval_sd.h**
- SD_CMD_APP_CMD : **stm32072b_eval_sd.c**
- SD_CMD_CLR_WRITE_PROT : **stm32072b_eval_sd.c**
- SD_CMD_ERASE : **stm32072b_eval_sd.c**
- SD_CMD_ERASE_GRP_END : **stm32072b_eval_sd.c**
- SD_CMD_ERASE_GRP_START : **stm32072b_eval_sd.c**
- SD_CMD_GO_IDLE_STATE : **stm32072b_eval_sd.c**
- SD_CMD_LENGTH : **stm32072b_eval_sd.c**
- SD_CMD_PROG_CSD : **stm32072b_eval_sd.c**
- SD_CMD_READ_MULT_BLOCK : **stm32072b_eval_sd.c**
- SD_CMD_READ_OCR : **stm32072b_eval_sd.c**
- SD_CMD_READ_SINGLE_BLOCK : **stm32072b_eval_sd.c**

- SD_CMD_SD_APP_OP_COND : **stm32072b_eval_sd.c**
- SD_CMD_SD_ERASE_GRP_END : **stm32072b_eval_sd.c**
- SD_CMD_SD_ERASE_GRP_START : **stm32072b_eval_sd.c**
- SD_CMD_SEND_CID : **stm32072b_eval_sd.c**
- SD_CMD_SEND_CSD : **stm32072b_eval_sd.c**
- SD_CMD_SEND_IF_COND : **stm32072b_eval_sd.c**
- SD_CMD_SEND_OP_COND : **stm32072b_eval_sd.c**
- SD_CMD_SEND_STATUS : **stm32072b_eval_sd.c**
- SD_CMD_SEND_WRITE_PROT : **stm32072b_eval_sd.c**
- SD_CMD_SET_BLOCK_COUNT : **stm32072b_eval_sd.c**
- SD_CMD_SET_BLOCKLEN : **stm32072b_eval_sd.c**
- SD_CMD_SET_WRITE_PROT : **stm32072b_eval_sd.c**
- SD_CMD_STOP_TRANSMISSION : **stm32072b_eval_sd.c**
- SD_CMD_UNTAG_ERASE_GROUP : **stm32072b_eval_sd.c**
- SD_CMD_UNTAG_SECTOR : **stm32072b_eval_sd.c**
- SD_CMD_WRITE_MULT_BLOCK : **stm32072b_eval_sd.c**
- SD_CMD_WRITE_SINGLE_BLOCK : **stm32072b_eval_sd.c**
- SD_CS_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- SD_CS_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- SD_CS_GPIO_PORT : **stm32072b_eval.h**
- SD_CS_HIGH : **stm32072b_eval.h**
- SD_CS_LOW : **stm32072b_eval.h**
- SD_CS_PIN : **stm32072b_eval.h**
- SD_CSD_STRUCT_V1 : **stm32072b_eval_sd.c**
- SD_CSD_STRUCT_V2 : **stm32072b_eval_sd.c**
- SD_DATA_CRC_ERROR : **stm32072b_eval_sd.c**
- SD_DATA_OK : **stm32072b_eval_sd.c**
- SD_DATA_OTHER_ERROR : **stm32072b_eval_sd.c**
- SD_DATA_WRITE_ERROR : **stm32072b_eval_sd.c**
- SD_DETECT_EXTI_IRQn : **stm32072b_eval.h**
- SD_DETECT_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- SD_DETECT_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- SD_DETECT_GPIO_PORT : **stm32072b_eval.h**
- SD_DETECT_PIN : **stm32072b_eval.h**
- SD_DUMMY_BYTE : **stm32072b_eval.c** , **stm32072b_eval_sd.c**
- SD_Error : **stm32072b_eval_sd.c**
- SD_GetCIDRegister() : **stm32072b_eval_sd.c**

- SD_GetCSDRegister() : **stm32072b_eval_sd.c**
- SD_GetDataResponse() : **stm32072b_eval_sd.c**
- SD_GoIdleState() : **stm32072b_eval_sd.c**
- SD_IO_CSState() : **stm32072b_eval.c** , **stm32072b_eval_sd.h**
- SD_IO_Init() : **stm32072b_eval_sd.h** , **stm32072b_eval.c**
- SD_IO_WriteByte() : **stm32072b_eval.c** , **stm32072b_eval_sd.h**
- SD_IO_WriteReadData() : **stm32072b_eval.c** , **stm32072b_eval_sd.h**
- SD_MAX_FRAME_LENGTH : **stm32072b_eval_sd.c**
- SD_MAX_TRY : **stm32072b_eval_sd.c**
- SD_NO_RESPONSE_EXPECTED : **stm32072b_eval.c**
- SD_NOT_PRESENT : **stm32072b_eval_sd.h**
- SD_PRESENT : **stm32072b_eval_sd.h**
- SD_R1_ADDRESS_ERROR : **stm32072b_eval_sd.c**
- SD_R1_COM_CRC_ERROR : **stm32072b_eval_sd.c**
- SD_R1_ERASE_RESET : **stm32072b_eval_sd.c**
- SD_R1_ERASE_SEQUENCE_ERROR : **stm32072b_eval_sd.c**
- SD_R1_ILLEGAL_COMMAND : **stm32072b_eval_sd.c**
- SD_R1_IN_IDLE_STATE : **stm32072b_eval_sd.c**
- SD_R1_NO_ERROR : **stm32072b_eval_sd.c**
- SD_R1_PARAMETER_ERROR : **stm32072b_eval_sd.c**
- SD_R2_CARD_ECC_FAILED : **stm32072b_eval_sd.c**
- SD_R2_CARD_LOCKED : **stm32072b_eval_sd.c**
- SD_R2_CC_ERROR : **stm32072b_eval_sd.c**
- SD_R2_ERASE_PARAM : **stm32072b_eval_sd.c**
- SD_R2_ERROR : **stm32072b_eval_sd.c**
- SD_R2_LOCKUNLOCK_ERROR : **stm32072b_eval_sd.c**
- SD_R2_NO_ERROR : **stm32072b_eval_sd.c**
- SD_R2_OUTOFRANGE : **stm32072b_eval_sd.c**
- SD_R2_WP_VIOLATION : **stm32072b_eval_sd.c**
- SD_ReadData() : **stm32072b_eval_sd.c**
- SD_SendCmd() : **stm32072b_eval_sd.c**
- SD_TOKEN_START_DATA_MULTIPLE_BLOCK_READ : **stm32072b_eval_sd.c**
- SD_TOKEN_START_DATA_MULTIPLE_BLOCK_WRITE : **stm32072b_eval_sd.c**
- SD_TOKEN_START_DATA_SINGLE_BLOCK_READ :

stm32072b_eval_sd.c

- SD_TOKEN_START_DATA_SINGLE_BLOCK_WRITE : **stm32072b_eval_sd.c**
- SD_TOKEN_STOP_DATA_MULTIPLE_BLOCK_WRITE : **stm32072b_eval_sd.c**
- SD_WaitData() : **stm32072b_eval_sd.c**
- SdStatus : **stm32072b_eval_sd.c**
- SEL_JOY_EXTI_IRQn : **stm32072b_eval.h**
- SEL_JOY_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- SEL_JOY_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- SEL_JOY_GPIO_PORT : **stm32072b_eval.h**
- SEL_JOY_PIN : **stm32072b_eval.h**
- SET_INDEX : **stm32072b_eval.c**
- SPIx_Error() : **stm32072b_eval.c**
- SPIx_FlushFifo() : **stm32072b_eval.c**
- SPIx_Init() : **stm32072b_eval.c**
- SPIx_MspInit() : **stm32072b_eval.c**
- SPIx_Read() : **stm32072b_eval.c**
- SPIx_Write() : **stm32072b_eval.c**
- SPIx_WriteReadData() : **stm32072b_eval.c**
- SpixTimeout : **stm32072b_eval.c**
- START_BYTE : **stm32072b_eval.c**

STM32072B_EVAL BSP User Manual

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

- t -

- TAMPER_BUTTON_EXTI_IRQn : **stm32072b_eval.h**
- TAMPER_BUTTON_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- TAMPER_BUTTON_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- TAMPER_BUTTON_GPIO_PORT : **stm32072b_eval.h**
- TAMPER_BUTTON_PIN : **stm32072b_eval.h**
- TAMPERx_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- TAMPERx_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- tsensor_drv : **stm32072b_eval_tsensor.c**
- TSENSOR_ERROR : **stm32072b_eval_tsensor.h**
- TSENSOR_I2C_ADDRESS_A01 : **stm32072b_eval_tsensor.h**
- TSENSOR_I2C_ADDRESS_A02 : **stm32072b_eval_tsensor.h**
- TSENSOR_IO_Init() : **stm32072b_eval.c**
- TSENSOR_IO_IsDeviceReady() : **stm32072b_eval.c**
- TSENSOR_IO_Read() : **stm32072b_eval.c**
- TSENSOR_IO_Write() : **stm32072b_eval.c**
- TSENSOR_MAX_TRIALS : **stm32072b_eval_tsensor.h**
- TSENSOR_OK : **stm32072b_eval_tsensor.h**
- TSENSOR_Status_TypDef : **stm32072b_eval_tsensor.h**
- TSENSORAddress : **stm32072b_eval_tsensor.c**

Generated on Wed Jul 5 2017 08:56:11 for STM32072B_EVAL BSP
User Manual by  1.7.6.1

STM32072B_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 : **stm32072b_eval.h**
- UP_JOY_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- UP_JOY_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- UP_JOY_GPIO_PORT : **stm32072b_eval.h**
- UP_JOY_PIN : **stm32072b_eval.h**

STM32072B_EVAL BSP User Manual

- b -

- BSP_COM_Init() : **stm32072b_eval.c**
- BSP_EEPROM_Init() : **stm32072b_eval_eeprom.c**
- BSP_EEPROM_ReadBuffer() : **stm32072b_eval_eeprom.c**
- BSP_EEPROM_TIMEOUT_UserCallback() : **stm32072b_eval_eeprom.c**
- BSP_EEPROM_WriteBuffer() : **stm32072b_eval_eeprom.c**
- BSP_GetVersion() : **stm32072b_eval.c**
- BSP_JOY_GetState() : **stm32072b_eval.c**
- BSP_JOY_Init() : **stm32072b_eval.c**
- BSP_LCD_Clear() : **stm32072b_eval_lcd.c**
- BSP_LCD_ClearStringLine() : **stm32072b_eval_lcd.c**
- BSP_LCD_DisplayChar() : **stm32072b_eval_lcd.c**
- BSP_LCD_DisplayOff() : **stm32072b_eval_lcd.c**
- BSP_LCD_DisplayOn() : **stm32072b_eval_lcd.c**
- BSP_LCD_DisplayStringAt() : **stm32072b_eval_lcd.c**
- BSP_LCD_DisplayStringAtLine() : **stm32072b_eval_lcd.c**
- BSP_LCD_DrawBitmap() : **stm32072b_eval_lcd.c**
- BSP_LCD_DrawCircle() : **stm32072b_eval_lcd.c**
- BSP_LCD_DrawEllipse() : **stm32072b_eval_lcd.c**
- BSP_LCD_DrawHLine() : **stm32072b_eval_lcd.c**

- BSP_LCD_DrawLine() : **stm32072b_eval_lcd.c**
- BSP_LCD_DrawPolygon() : **stm32072b_eval_lcd.c**
- BSP_LCD_DrawRect() : **stm32072b_eval_lcd.c**
- BSP_LCD_DrawVLine() : **stm32072b_eval_lcd.c**
- BSP_LCD_FillCircle() : **stm32072b_eval_lcd.c**
- BSP_LCD_FillEllipse() : **stm32072b_eval_lcd.c**
- BSP_LCD_FillRect() : **stm32072b_eval_lcd.c**
- BSP_LCD_GetBackColor() : **stm32072b_eval_lcd.c**
- BSP_LCD_GetFont() : **stm32072b_eval_lcd.c**
- BSP_LCD_GetTextColor() : **stm32072b_eval_lcd.c**
- BSP_LCD_GetXSize() : **stm32072b_eval_lcd.c**
- BSP_LCD_GetYSize() : **stm32072b_eval_lcd.c**
- BSP_LCD_Init() : **stm32072b_eval_lcd.c**
- BSP_LCD_ReadPixel() : **stm32072b_eval_lcd.c**
- BSP_LCD_SetBackColor() : **stm32072b_eval_lcd.c** , **stm32072b_eval_lcd.h**
- BSP_LCD_SetFont() : **stm32072b_eval_lcd.c**
- BSP_LCD_SetTextColor() : **stm32072b_eval_lcd.c** , **stm32072b_eval_lcd.h**
- BSP_LED_Init() : **stm32072b_eval.c**
- BSP_LED_Off() : **stm32072b_eval.c**
- BSP_LED_On() : **stm32072b_eval.c**
- BSP_LED_Toggle() : **stm32072b_eval.c**
- BSP_PB_GetState() : **stm32072b_eval.c**
- BSP_PB_Init() : **stm32072b_eval.c**
- BSP_SD_Erase() : **stm32072b_eval_sd.c**
- BSP_SD_GetCardInfo() : **stm32072b_eval_sd.c**
- BSP_SD_GetStatus() : **stm32072b_eval_sd.c**
- BSP_SD_Init() : **stm32072b_eval_sd.c**
- BSP_SD_IsDetected() : **stm32072b_eval_sd.c**
- BSP_SD_ReadBlocks() : **stm32072b_eval_sd.c**
- BSP_SD_WriteBlocks() : **stm32072b_eval_sd.c**
- BSP_TSENSOR_Init() : **stm32072b_eval_tsensor.c**
- BSP_TSENSOR_ReadStatus() : **stm32072b_eval_tsensor.c**
- BSP_TSENSOR_ReadTemp() : **stm32072b_eval_tsensor.c**

- e -

- EEPROM_I2C_Init() : **stm32072b_eval_eeprom.c**
- EEPROM_I2C_ReadBuffer() : **stm32072b_eval_eeprom.c**
- EEPROM_I2C_WaitEepromStandbyState() : **stm32072b_eval_eeprom.c**
- EEPROM_I2C_WritePage() : **stm32072b_eval_eeprom.c**
- EEPROM_IO_Init() : **stm32072b_eval.c** , **stm32072b_eval_eeprom.h**
- EEPROM_IO_IsDeviceReady() : **stm32072b_eval_eeprom.h** , **stm32072b_eval.c**
- EEPROM_IO_ReadData() : **stm32072b_eval_eeprom.h** , **stm32072b_eval.c**
- EEPROM_IO_WriteData() : **stm32072b_eval.c** , **stm32072b_eval_eeprom.h**

- h -

- HDMI_CEC_IO_Init() : **stm32072b_eval.c**
- HDMI_CEC_IO_ReadData() : **stm32072b_eval.c**
- HDMI_CEC_IO_WriteData() : **stm32072b_eval.c**

- i -

- I2C1_Error() : **stm32072b_eval.c**
- I2C1_Init() : **stm32072b_eval.c**
- I2C1_IsDeviceReady() : **stm32072b_eval.c**
- I2C1_MspInit() : **stm32072b_eval.c**
- I2C1_ReadBuffer() : **stm32072b_eval.c**
- I2C1_TransmitData() : **stm32072b_eval.c**
- I2C1_WriteBuffer() : **stm32072b_eval.c**
- I2C2_Error() : **stm32072b_eval.c**
- I2C2_Init() : **stm32072b_eval.c**
- I2C2_MspInit() : **stm32072b_eval.c**
- I2C2_ReceiveData() : **stm32072b_eval.c**

- l -

- LCD_Delay() : **stm32072b_eval.c**
- LCD_DrawChar() : **stm32072b_eval_lcd.c**

- LCD_DrawPixel() : **stm32072b_eval_lcd.c**
- LCD_IO_Init() : **stm32072b_eval.c**
- LCD_IO_ReadData() : **stm32072b_eval.c**
- LCD_IO_WriteMultipleData() : **stm32072b_eval.c**
- LCD_IO_WriteReg() : **stm32072b_eval.c**
- LCD_SetDisplayWindow() : **stm32072b_eval_lcd.c**

- s -

- SD_GetCIDRegister() : **stm32072b_eval_sd.c**
- SD_GetCSDRegister() : **stm32072b_eval_sd.c**
- SD_GetDataResponse() : **stm32072b_eval_sd.c**
- SD_GoIdleState() : **stm32072b_eval_sd.c**
- SD_IO_CSState() : **stm32072b_eval.c** , **stm32072b_eval_sd.h**
- SD_IO_Init() : **stm32072b_eval_sd.h** , **stm32072b_eval.c**
- SD_IO_WriteByte() : **stm32072b_eval.c** , **stm32072b_eval_sd.h**
- SD_IO_WriteReadData() : **stm32072b_eval.c** , **stm32072b_eval_sd.h**
- SD_ReadData() : **stm32072b_eval_sd.c**
- SD_SendCmd() : **stm32072b_eval_sd.c**
- SD_WaitData() : **stm32072b_eval_sd.c**
- SPIx_Error() : **stm32072b_eval.c**
- SPIx_FlushFifo() : **stm32072b_eval.c**
- SPIx_Init() : **stm32072b_eval.c**
- SPIx_MspInit() : **stm32072b_eval.c**
- SPIx_Read() : **stm32072b_eval.c**
- SPIx_Write() : **stm32072b_eval.c**
- SPIx_WriteReadData() : **stm32072b_eval.c**

- t -

- TSENSOR_IO_Init() : **stm32072b_eval.c**
- TSENSOR_IO_IsDeviceReady() : **stm32072b_eval.c**
- TSENSOR_IO_Read() : **stm32072b_eval.c**
- TSENSOR_IO_Write() : **stm32072b_eval.c**

User Manual by  1.7.6.1

STM32072B_EVAL BSP User Manual

- b -

- bitmap : **stm32072b_eval_lcd.c**
- BUTTON_IRQn : **stm32072b_eval.c**
- BUTTON_PIN : **stm32072b_eval.c**
- BUTTON_PORT : **stm32072b_eval.c**

- c -

- COM_RX_AF : **stm32072b_eval.c**
- COM_RX_PIN : **stm32072b_eval.c**
- COM_RX_PORT : **stm32072b_eval.c**
- COM_TX_AF : **stm32072b_eval.c**
- COM_TX_PIN : **stm32072b_eval.c**
- COM_TX_PORT : **stm32072b_eval.c**
- COM_USART : **stm32072b_eval.c**

- d -

- DrawProp : **stm32072b_eval_lcd.c**

- e -

- EEPROM_I2C_Drv : **stm32072b_eval_eeprom.c**
- EEPROM_SelectedDevice : **stm32072b_eval_eeprom.c**
- EEPROMAddress : **stm32072b_eval_eeprom.c**
- EEPROMDataRead : **stm32072b_eval_eeprom.c**
- EEPROMDataWrite : **stm32072b_eval_eeprom.c**
- EEPROMPageSize : **stm32072b_eval_eeprom.c**

- f -

- flag_SDHC : **stm32072b_eval_sd.c**

- h -

- heval_I2c1 : **stm32072b_eval.c**
- heval_I2c2 : **stm32072b_eval.c**
- heval_Spi : **stm32072b_eval.c**

- i -

- I2c1Timeout : **stm32072b_eval.c**
- I2c2Timeout : **stm32072b_eval.c**

- j -

- JOY_IRQn : **stm32072b_eval.c**
- JOY_PIN : **stm32072b_eval.c**
- JOY_PORT : **stm32072b_eval.c**

- l -

- lcd_drv : **stm32072b_eval_lcd.c**
- LED_PIN : **stm32072b_eval.c**
- LED_PORT : **stm32072b_eval.c**

- s -

- SdStatus : **stm32072b_eval_sd.c**
- SpixTimeout : **stm32072b_eval.c**

- t -

- tsensor_drv : **stm32072b_eval_tsensor.c**
- TSENSORAddress : **stm32072b_eval_tsensor.c**

STM32072B_EVAL BSP User Manual

- pPoint : **stm32072b_eval_lcd.h**

Generated on Wed Jul 5 2017 08:56:11 for STM32072B_EVAL BSP User Manual by doxygen 1.7.6.1

STM32072B_EVAL BSP User Manual

- Button_TypeDef : **stm32072b_eval.h**
- ButtonMode_TypeDef : **stm32072b_eval.h**
- COM_TypeDef : **stm32072b_eval.h**
- JOYMode_TypeDef : **stm32072b_eval.h**
- JOYState_TypeDef : **stm32072b_eval.h**
- Led_TypeDef : **stm32072b_eval.h**
- Line_ModeTypdef : **stm32072b_eval_lcd.h**
- SD_Answer_type : **stm32072b_eval_sd.c**
- SD_Error : **stm32072b_eval_sd.c**
- TSENSOR_Status_TypDef : **stm32072b_eval_tsensor.h**

STM32072B_EVAL BSP User Manual

- b -

- BSP_SD_ERROR : **stm32072b_eval_sd.h**
- BSP_SD_OK : **stm32072b_eval_sd.h**
- BSP_SD_TIMEOUT : **stm32072b_eval_sd.h**
- BUTTON_MODE_EXTI : **stm32072b_eval.h**
- BUTTON_MODE_GPIO : **stm32072b_eval.h**
- BUTTON_TAMPER : **stm32072b_eval.h**

- c -

- CENTER_MODE : **stm32072b_eval_lcd.h**
- COM1 : **stm32072b_eval.h**

- j -

- JOY_DOWN : **stm32072b_eval.h**
- JOY_LEFT : **stm32072b_eval.h**
- JOY_MODE_EXTI : **stm32072b_eval.h**
- JOY_MODE_GPIO : **stm32072b_eval.h**
- JOY_NONE : **stm32072b_eval.h**
- JOY_RIGHT : **stm32072b_eval.h**

- JOY_SEL : **stm32072b_eval.h**
- JOY_UP : **stm32072b_eval.h**

- l -

- LED1 : **stm32072b_eval.h**
- LED2 : **stm32072b_eval.h**
- LED3 : **stm32072b_eval.h**
- LED4 : **stm32072b_eval.h**
- LED_BLUE : **stm32072b_eval.h**
- LED_GREEN : **stm32072b_eval.h**
- LED_ORANGE : **stm32072b_eval.h**
- LED_RED : **stm32072b_eval.h**
- LEFT_MODE : **stm32072b_eval_lcd.h**

- m -

- MSD_ERROR : **stm32072b_eval_sd.h**
- MSD_OK : **stm32072b_eval_sd.h**

- r -

- RIGHT_MODE : **stm32072b_eval_lcd.h**

- s -

- SD_ANSWER_R1_EXPECTED : **stm32072b_eval_sd.c**
- SD_ANSWER_R1B_EXPECTED : **stm32072b_eval_sd.c**
- SD_ANSWER_R2_EXPECTED : **stm32072b_eval_sd.c**
- SD_ANSWER_R3_EXPECTED : **stm32072b_eval_sd.c**
- SD_ANSWER_R4R5_EXPECTED : **stm32072b_eval_sd.c**
- SD_ANSWER_R7_EXPECTED : **stm32072b_eval_sd.c**
- SD_DATA_CRC_ERROR : **stm32072b_eval_sd.c**
- SD_DATA_OK : **stm32072b_eval_sd.c**
- SD_DATA_OTHER_ERROR : **stm32072b_eval_sd.c**
- SD_DATA_WRITE_ERROR : **stm32072b_eval_sd.c**
- SD_R1_ADDRESS_ERROR : **stm32072b_eval_sd.c**
- SD_R1_COM_CRC_ERROR : **stm32072b_eval_sd.c**

- SD_R1_ERASE_RESET : **stm32072b_eval_sd.c**
- SD_R1_ERASE_SEQUENCE_ERROR : **stm32072b_eval_sd.c**
- SD_R1_ILLEGAL_COMMAND : **stm32072b_eval_sd.c**
- SD_R1_IN_IDLE_STATE : **stm32072b_eval_sd.c**
- SD_R1_NO_ERROR : **stm32072b_eval_sd.c**
- SD_R1_PARAMETER_ERROR : **stm32072b_eval_sd.c**
- SD_R2_CARD_ECC_FAILED : **stm32072b_eval_sd.c**
- SD_R2_CARD_LOCKED : **stm32072b_eval_sd.c**
- SD_R2_CC_ERROR : **stm32072b_eval_sd.c**
- SD_R2_ERASE_PARAM : **stm32072b_eval_sd.c**
- SD_R2_ERROR : **stm32072b_eval_sd.c**
- SD_R2_LOCKUNLOCK_ERROR : **stm32072b_eval_sd.c**
- SD_R2_NO_ERROR : **stm32072b_eval_sd.c**
- SD_R2_OUTOFRANGE : **stm32072b_eval_sd.c**
- SD_R2_WP_VIOLATION : **stm32072b_eval_sd.c**

- t -

- TSENSOR_ERROR : **stm32072b_eval_tsensor.h**
- TSENSOR_OK : **stm32072b_eval_tsensor.h**

STM32072B_EVAL BSP User Manual

- _ -

- __STM32072B_EVAL_BSP_VERSION : **stm32072b_eval.c**
- __STM32072B_EVAL_BSP_VERSION_MAIN : **stm32072b_eval.c**
- __STM32072B_EVAL_BSP_VERSION_RC : **stm32072b_eval.c**
- __STM32072B_EVAL_BSP_VERSION_SUB1 : **stm32072b_eval.c**
- __STM32072B_EVAL_BSP_VERSION_SUB2 : **stm32072b_eval.c**

STM32072B_EVAL BSP User Manual

- a -

- ABS : **stm32072b_eval_lcd.c**

STM32072B_EVAL BSP User Manual

- b -

- BSP_EEPROM_M24LR64 : **stm32072b_eval_eeprom.h**
- BUTTONn : **stm32072b_eval.h**

STM32072B_EVAL BSP User Manual

- c -

- COMn : **stm32072b_eval.h**
- COMx_CLK_DISABLE : **stm32072b_eval.h**
- COMx_CLK_ENABLE : **stm32072b_eval.h**
- COMx_CTS_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- COMx_CTS_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- COMx_RTS_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- COMx_RTS_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- COMx_RX_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- COMx_RX_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- COMx_TX_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- COMx_TX_GPIO_CLK_ENABLE : **stm32072b_eval.h**

STM32072B_EVAL BSP User Manual

- d -

- DOWN_JOY_EXTI_IRQn : **stm32072b_eval.h**
- DOWN_JOY_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- DOWN_JOY_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- DOWN_JOY_GPIO_PORT : **stm32072b_eval.h**
- DOWN_JOY_PIN : **stm32072b_eval.h**

STM32072B_EVAL BSP User Manual

- e -

- EEPROM_ADDRESS_M24LR64_A01 : **stm32072b_eval_eeprom.h**
- EEPROM_ADDRESS_M24LR64_A02 : **stm32072b_eval_eeprom.h**
- EEPROM_FAIL : **stm32072b_eval_eeprom.h**
- EEPROM_MAX_TRIALS : **stm32072b_eval_eeprom.h**
- EEPROM_OK : **stm32072b_eval_eeprom.h**
- EEPROM_PAGESIZE_M24LR64 : **stm32072b_eval_eeprom.h**
- EEPROM_TIMEOUT : **stm32072b_eval_eeprom.h**
- EVAL_COM1 : **stm32072b_eval.h**
- EVAL_COM1_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_COM1_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_COM1_CTS_AF : **stm32072b_eval.h**
- EVAL_COM1_CTS_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_COM1_CTS_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_COM1_CTS_GPIO_PORT : **stm32072b_eval.h**
- EVAL_COM1_CTS_PIN : **stm32072b_eval.h**
- EVAL_COM1_IRQn : **stm32072b_eval.h**
- EVAL_COM1_RTS_AF : **stm32072b_eval.h**
- EVAL_COM1_RTS_GPIO_CLK_DISABLE : **stm32072b_eval.h**

- EVAL_COM1_RTS_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_COM1_RTS_GPIO_PORT : **stm32072b_eval.h**
- EVAL_COM1_RTS_PIN : **stm32072b_eval.h**
- EVAL_COM1_RX_AF : **stm32072b_eval.h**
- EVAL_COM1_RX_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_COM1_RX_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_COM1_RX_GPIO_PORT : **stm32072b_eval.h**
- EVAL_COM1_RX_PIN : **stm32072b_eval.h**
- EVAL_COM1_TX_AF : **stm32072b_eval.h**
- EVAL_COM1_TX_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_COM1_TX_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_COM1_TX_GPIO_PORT : **stm32072b_eval.h**
- EVAL_COM1_TX_PIN : **stm32072b_eval.h**
- EVAL_I2C1 : **stm32072b_eval.h**
- EVAL_I2C1_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_I2C1_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_I2C1_FORCE_RESET : **stm32072b_eval.h**
- EVAL_I2C1_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_I2C1_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_I2C1_GPIO_PORT : **stm32072b_eval.h**
- EVAL_I2C1_RELEASE_RESET : **stm32072b_eval.h**
- EVAL_I2C1_SCL_PIN : **stm32072b_eval.h**
- EVAL_I2C1_SCL_SDA_AF : **stm32072b_eval.h**
- EVAL_I2C1_SDA_PIN : **stm32072b_eval.h**
- EVAL_I2C1_TIMEOUT_MAX : **stm32072b_eval.h**
- EVAL_I2C2 : **stm32072b_eval.h**
- EVAL_I2C2_AF : **stm32072b_eval.h**
- EVAL_I2C2_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_I2C2_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_I2C2_FORCE_RESET : **stm32072b_eval.h**
- EVAL_I2C2_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_I2C2_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_I2C2_GPIO_PORT : **stm32072b_eval.h**
- EVAL_I2C2_IRQn : **stm32072b_eval.h**
- EVAL_I2C2_RELEASE_RESET : **stm32072b_eval.h**
- EVAL_I2C2_SCL_PIN : **stm32072b_eval.h**
- EVAL_I2C2_SDA_PIN : **stm32072b_eval.h**

- EVAL_I2C2_TIMEOUT_MAX : **stm32072b_eval.h**
- EVAL_SPIx : **stm32072b_eval.h**
- EVAL_SPIx_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_SPIx_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_SPIx_FORCE_RESET : **stm32072b_eval.h**
- EVAL_SPIx_MISO_AF : **stm32072b_eval.h**
- EVAL_SPIx_MISO_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_SPIx_MISO_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_SPIx_MISO_GPIO_PORT : **stm32072b_eval.h**
- EVAL_SPIx_MISO_PIN : **stm32072b_eval.h**
- EVAL_SPIx_MOSI_AF : **stm32072b_eval.h**
- EVAL_SPIx_MOSI_DIR_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_SPIx_MOSI_DIR_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_SPIx_MOSI_DIR_GPIO_PORT : **stm32072b_eval.h**
- EVAL_SPIx_MOSI_DIR_PIN : **stm32072b_eval.h**
- EVAL_SPIx_MOSI_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_SPIx_MOSI_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_SPIx_MOSI_GPIO_PORT : **stm32072b_eval.h**
- EVAL_SPIx_MOSI_PIN : **stm32072b_eval.h**
- EVAL_SPIx_RELEASE_RESET : **stm32072b_eval.h**
- EVAL_SPIx_SCK_AF : **stm32072b_eval.h**
- EVAL_SPIx_SCK_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- EVAL_SPIx_SCK_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- EVAL_SPIx_SCK_GPIO_PORT : **stm32072b_eval.h**
- EVAL_SPIx_SCK_PIN : **stm32072b_eval.h**
- EVAL_SPIx_TIMEOUT_MAX : **stm32072b_eval.h**

STM32072B_EVAL BSP User Manual

- h -

- HDMI_CEC_HPD_SINK_CLK_DISABLE : **stm32072b_eval.h**
- HDMI_CEC_HPD_SINK_CLK_ENABLE : **stm32072b_eval.h**
- HDMI_CEC_HPD_SINK_GPIO_PORT : **stm32072b_eval.h**
- HDMI_CEC_HPD_SINK_PIN : **stm32072b_eval.h**
- HDMI_CEC_HPD_SOURCE_CLK_DISABLE : **stm32072b_eval.h**
- HDMI_CEC_HPD_SOURCE_CLK_ENABLE : **stm32072b_eval.h**
- HDMI_CEC_HPD_SOURCE_GPIO_PORT : **stm32072b_eval.h**
- HDMI_CEC_HPD_SOURCE_PIN : **stm32072b_eval.h**
- HDMI_CEC_I2C_ADDRESS : **stm32072b_eval.h**
- HDMI_CEC_IRQn : **stm32072b_eval.h**
- HDMI_CEC_LINE_AF : **stm32072b_eval.h**
- HDMI_CEC_LINE_CLK_DISABLE : **stm32072b_eval.h**
- HDMI_CEC_LINE_CLK_ENABLE : **stm32072b_eval.h**
- HDMI_CEC_LINE_GPIO_PORT : **stm32072b_eval.h**
- HDMI_CEC_LINE_PIN : **stm32072b_eval.h**

STM32072B_EVAL BSP User Manual

- i -

- I2C1_TIMING : **stm32072b_eval.h**
- I2C2_TIMING : **stm32072b_eval.h**

STM32072B_EVAL BSP User Manual

- j -

- JOYn : **stm32072b_eval.h**
- JOYx_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- JOYx_GPIO_CLK_ENABLE : **stm32072b_eval.h**

STM32072B_EVAL BSP User Manual

- l -

- LCD_COLOR_BLACK : **stm32072b_eval_lcd.h**
- LCD_COLOR_BLUE : **stm32072b_eval_lcd.h**
- LCD_COLOR_BROWN : **stm32072b_eval_lcd.h**
- LCD_COLOR_CYAN : **stm32072b_eval_lcd.h**
- LCD_COLOR_DARKBLUE : **stm32072b_eval_lcd.h**
- LCD_COLOR_DARKCYAN : **stm32072b_eval_lcd.h**
- LCD_COLOR_DARKGRAY : **stm32072b_eval_lcd.h**
- LCD_COLOR_DARKGREEN : **stm32072b_eval_lcd.h**
- LCD_COLOR_DARKMAGENTA : **stm32072b_eval_lcd.h**
- LCD_COLOR_DARKRED : **stm32072b_eval_lcd.h**
- LCD_COLOR_DARKYELLOW : **stm32072b_eval_lcd.h**
- LCD_COLOR_GRAY : **stm32072b_eval_lcd.h**
- LCD_COLOR_GREEN : **stm32072b_eval_lcd.h**
- LCD_COLOR_LIGHTBLUE : **stm32072b_eval_lcd.h**
- LCD_COLOR_LIGHTCYAN : **stm32072b_eval_lcd.h**
- LCD_COLOR_LIGHTGRAY : **stm32072b_eval_lcd.h**
- LCD_COLOR_LIGHTGREEN : **stm32072b_eval_lcd.h**
- LCD_COLOR_LIGHTMAGENTA : **stm32072b_eval_lcd.h**
- LCD_COLOR_LIGHTRED : **stm32072b_eval_lcd.h**
- LCD_COLOR_LIGHTYELLOW : **stm32072b_eval_lcd.h**

- LCD_COLOR_MAGENTA : **stm32072b_eval_lcd.h**
- LCD_COLOR_ORANGE : **stm32072b_eval_lcd.h**
- LCD_COLOR_RED : **stm32072b_eval_lcd.h**
- LCD_COLOR_WHITE : **stm32072b_eval_lcd.h**
- LCD_COLOR_YELLOW : **stm32072b_eval_lcd.h**
- LCD_CS_HIGH : **stm32072b_eval.h**
- LCD_CS_LOW : **stm32072b_eval.h**
- LCD_DEFAULT_FONT : **stm32072b_eval_lcd.h**
- LCD_ERROR : **stm32072b_eval_lcd.h**
- LCD_NCS_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- LCD_NCS_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- LCD_NCS_GPIO_PORT : **stm32072b_eval.h**
- LCD_NCS_PIN : **stm32072b_eval.h**
- LCD_OK : **stm32072b_eval_lcd.h**
- LCD_READ_REG : **stm32072b_eval.c**
- LCD_TIMEOUT : **stm32072b_eval_lcd.h**
- LCD_WRITE_REG : **stm32072b_eval.c**
- LED1_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- LED1_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- LED1_GPIO_PORT : **stm32072b_eval.h**
- LED1_PIN : **stm32072b_eval.h**
- LED2_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- LED2_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- LED2_GPIO_PORT : **stm32072b_eval.h**
- LED2_PIN : **stm32072b_eval.h**
- LED3_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- LED3_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- LED3_GPIO_PORT : **stm32072b_eval.h**
- LED3_PIN : **stm32072b_eval.h**
- LED4_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- LED4_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- LED4_GPIO_PORT : **stm32072b_eval.h**
- LED4_PIN : **stm32072b_eval.h**
- LEDn : **stm32072b_eval.h**
- LEDx_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- LEDx_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- LEFT_JOY_EXTI_IRQn : **stm32072b_eval.h**

- LEFT_JOY_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- LEFT_JOY_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- LEFT_JOY_GPIO_PORT : **stm32072b_eval.h**
- LEFT_JOY_PIN : **stm32072b_eval.h**

STM32072B_EVAL BSP User Manual

- m -

- MAX_HEIGHT_FONT : **stm32072b_eval_lcd.c**
- MAX_WIDTH_FONT : **stm32072b_eval_lcd.c**

STM32072B_EVAL BSP User Manual

- o -

- OFFSET_BITMAP : **stm32072b_eval_lcd.c**

STM32072B_EVAL BSP User Manual

- p -

- POLY_X : **stm32072b_eval_lcd.c**
- POLY_Y : **stm32072b_eval_lcd.c**

STM32072B_EVAL BSP User Manual

- r -

- READ_STATUS : **stm32072b_eval.c**
- RIGHT_JOY_EXTI_IRQn : **stm32072b_eval.h**
- RIGHT_JOY_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- RIGHT_JOY_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- RIGHT_JOY_GPIO_PORT : **stm32072b_eval.h**
- RIGHT_JOY_PIN : **stm32072b_eval.h**

STM32072B_EVAL BSP User Manual

- s -

- SD_BLOCK_SIZE : **stm32072b_eval_sd.h**
- SD_CMD_APP_CMD : **stm32072b_eval_sd.c**
- SD_CMD_CLR_WRITE_PROT : **stm32072b_eval_sd.c**
- SD_CMD_ERASE : **stm32072b_eval_sd.c**
- SD_CMD_ERASE_GRP_END : **stm32072b_eval_sd.c**
- SD_CMD_ERASE_GRP_START : **stm32072b_eval_sd.c**
- SD_CMD_GO_IDLE_STATE : **stm32072b_eval_sd.c**
- SD_CMD_LENGTH : **stm32072b_eval_sd.c**
- SD_CMD_PROG_CSD : **stm32072b_eval_sd.c**
- SD_CMD_READ_MULT_BLOCK : **stm32072b_eval_sd.c**
- SD_CMD_READ_OCR : **stm32072b_eval_sd.c**
- SD_CMD_READ_SINGLE_BLOCK : **stm32072b_eval_sd.c**
- SD_CMD_SD_APP_OP_COND : **stm32072b_eval_sd.c**
- SD_CMD_SD_ERASE_GRP_END : **stm32072b_eval_sd.c**
- SD_CMD_SD_ERASE_GRP_START : **stm32072b_eval_sd.c**
- SD_CMD_SEND_CID : **stm32072b_eval_sd.c**
- SD_CMD_SEND_CSD : **stm32072b_eval_sd.c**
- SD_CMD_SEND_IF_COND : **stm32072b_eval_sd.c**
- SD_CMD_SEND_OP_COND : **stm32072b_eval_sd.c**
- SD_CMD_SEND_STATUS : **stm32072b_eval_sd.c**

- SD_CMD_SEND_WRITE_PROT : **stm32072b_eval_sd.c**
- SD_CMD_SET_BLOCK_COUNT : **stm32072b_eval_sd.c**
- SD_CMD_SET_BLOCKLEN : **stm32072b_eval_sd.c**
- SD_CMD_SET_WRITE_PROT : **stm32072b_eval_sd.c**
- SD_CMD_STOP_TRANSMISSION : **stm32072b_eval_sd.c**
- SD_CMD_UNTAG_ERASE_GROUP : **stm32072b_eval_sd.c**
- SD_CMD_UNTAG_SECTOR : **stm32072b_eval_sd.c**
- SD_CMD_WRITE_MULT_BLOCK : **stm32072b_eval_sd.c**
- SD_CMD_WRITE_SINGLE_BLOCK : **stm32072b_eval_sd.c**
- SD_CS_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- SD_CS_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- SD_CS_GPIO_PORT : **stm32072b_eval.h**
- SD_CS_HIGH : **stm32072b_eval.h**
- SD_CS_LOW : **stm32072b_eval.h**
- SD_CS_PIN : **stm32072b_eval.h**
- SD_CSD_STRUCT_V1 : **stm32072b_eval_sd.c**
- SD_CSD_STRUCT_V2 : **stm32072b_eval_sd.c**
- SD_DETECT_EXTI_IRQn : **stm32072b_eval.h**
- SD_DETECT_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- SD_DETECT_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- SD_DETECT_GPIO_PORT : **stm32072b_eval.h**
- SD_DETECT_PIN : **stm32072b_eval.h**
- SD_DUMMY_BYTE : **stm32072b_eval.c** , **stm32072b_eval_sd.c**
- SD_MAX_FRAME_LENGTH : **stm32072b_eval_sd.c**
- SD_MAX_TRY : **stm32072b_eval_sd.c**
- SD_NO_RESPONSE_EXPECTED : **stm32072b_eval.c**
- SD_NOT_PRESENT : **stm32072b_eval_sd.h**
- SD_PRESENT : **stm32072b_eval_sd.h**
- SD_TOKEN_START_DATA_MULTIPLE_BLOCK_READ : **stm32072b_eval_sd.c**
- SD_TOKEN_START_DATA_MULTIPLE_BLOCK_WRITE : **stm32072b_eval_sd.c**
- SD_TOKEN_START_DATA_SINGLE_BLOCK_READ : **stm32072b_eval_sd.c**
- SD_TOKEN_START_DATA_SINGLE_BLOCK_WRITE : **stm32072b_eval_sd.c**
- SD_TOKEN_STOP_DATA_MULTIPLE_BLOCK_WRITE :

stm32072b_eval_sd.c

- SEL_JOY_EXTI_IRQn : **stm32072b_eval.h**
- SEL_JOY_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- SEL_JOY_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- SEL_JOY_GPIO_PORT : **stm32072b_eval.h**
- SEL_JOY_PIN : **stm32072b_eval.h**
- SET_INDEX : **stm32072b_eval.c**
- START_BYTE : **stm32072b_eval.c**

STM32072B_EVAL BSP User Manual

- t -

- TAMPER_BUTTON_EXTI_IRQn : **stm32072b_eval.h**
- TAMPER_BUTTON_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- TAMPER_BUTTON_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- TAMPER_BUTTON_GPIO_PORT : **stm32072b_eval.h**
- TAMPER_BUTTON_PIN : **stm32072b_eval.h**
- TAMPERx_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- TAMPERx_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- TSENSOR_I2C_ADDRESS_A01 : **stm32072b_eval_tsensor.h**
- TSENSOR_I2C_ADDRESS_A02 : **stm32072b_eval_tsensor.h**
- TSENSOR_MAX_TRIALS : **stm32072b_eval_tsensor.h**

STM32072B_EVAL BSP User Manual

- u -

- UP_JOY_EXTI_IRQn : **stm32072b_eval.h**
- UP_JOY_GPIO_CLK_DISABLE : **stm32072b_eval.h**
- UP_JOY_GPIO_CLK_ENABLE : **stm32072b_eval.h**
- UP_JOY_GPIO_PORT : **stm32072b_eval.h**
- UP_JOY_PIN : **stm32072b_eval.h**

STM32072B_EVAL BSP User Manual

stm32072b_eval.c File Reference

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

```
#include "stm32072b_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	**__STM32072B_EVAL_BSP_VERSION_MAIN** (0x02) STM32072B EVAL BSP Driver version number V2.1.8.
#define	**__STM32072B_EVAL_BSP_VERSION_SUB1** (0x01)
#define	**__STM32072B_EVAL_BSP_VERSION_SUB2** (0x08)
#define	**__STM32072B_EVAL_BSP_VERSION_RC** (0x00)
#define	**__STM32072B_EVAL_BSP_VERSION**

Functions

static void	**I2C1_Init** (void) I2C Bus initialization.
static void	**I2C1_Error** (void) Manages error callback by re-initializing I2C.
static void	**I2C1_MspInit** (I2C_HandleTypeDef *hi2c) I2C MSP Initialization.
static HAL_StatusTypeDef	**I2C1_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 HAL_StatusTypeDef	**I2C1_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	**I2C1_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.
static HAL_StatusTypeDef	**I2C1_TransmitData** (uint8_t *pBuffer, uint16_t Length) Write buffer through I2C.
static void	**I2C2_Init** (void) I2C Bus initialization.
static void	**I2C2_Error** (void) Discovery I2C2 error treatment function.
static void	**I2C2_MspInit** (I2C_HandleTypeDef *hi2c) I2C MSP Initialization.
static HAL_StatusTypeDef	**I2C2_ReceiveData** (uint16_t Addr, uint8_t *pBuffer, uint16_t Length)

	Read a register of the device through I2C.
void	**EEPROM_IO_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_IO_WriteData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Write data to I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_IO_ReadData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Read data from I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_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	**HDMI_CEC_IO_Init** (void) Initializes CEC low level.
HAL_StatusTypeDef	**HDMI_CEC_IO_WriteData** (uint8_t *pBuffer, uint16_t BufferSize)

	Write data to I2C HDMI CEC driver.
HAL_StatusTypeDef	**HDMI_CEC_IO_ReadData** (uint16_t DevAddress, uint8_t *pBuffer, uint16_t BufferSize) Read data to I2C HDMI CEC driver.
static void	**SPIx_Init** (void) SPIx Bus initialization.
static void	**SPIx_Write** (uint8_t Value) SPI Write a byte to device.
static void	**SPIx_WriteReadData** (const uint8_t *DataIn, uint8_t *DataOut, uint16_t DataLegnth) SPI Write a byte to device.
static void	**SPIx_FlushFifo** (void) SPIx_FlushFifo.
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) SPI MSP Init.
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) Writes address on LCD register.
uint16_t	**LCD_IO_ReadData** (uint16_t Reg) Read data from LCD data register.
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_CSState** (uint8_t state)
void	**SD_IO_WriteReadData** (const uint8_t *DataIn, uint8_t *DataOut, uint16_t DataLength) Write a byte on the SD.
uint8_t	**SD_IO_WriteByte** (uint8_t Data) Writes a byte on the SD.
uint32_t	**BSP_GetVersion** (void) This method returns the STM32F072B 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** Mode) Configures Tamper Button GPIO or EXTI Line.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
uint8_t	**BSP_JOY_Init** (**JOYMode_TypeDef** Joy_Mode) Configures joystick GPIO and EXTI modes.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the current joystick status.
void	**BSP_COM_Init** (**COM_TypeDef** COM,

UART_HandleTypeDef *huart)
Configures COM port.

Variables

GPIO_TypeDef *	**LED_PORT** [**LEDn**] LED variables.
const uint16_t	**LED_PIN** [**LEDn**]
GPIO_TypeDef *	**BUTTON_PORT** [**BUTTONn**] = {**TAMPER_BUTTON_GPIO_PORT**} BUTTON variables.
const uint16_t	**BUTTON_PIN** [**BUTTONn**] = {**TAMPER_BUTTON_PIN**}
const uint8_t	**BUTTON_IRQn** [**BUTTONn**] = {**TAMPER_BUTTON_EXTI_IRQn**}
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	**I2c1Timeout** = **EVAL_I2C1_TIMEOUT_MAX** BUS variables.
	I2c2Timeout =

uint32_t	**EVAL_I2C2_TIMEOUT_MAX**
I2C_HandleTypeDef	**heval_I2c1**
I2C_HandleTypeDef	**heval_I2c2**
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 ports.

Author:
MCD Application Team

Attention:

Definition in file **stm32072b_eval.c**.

STM32072B_EVAL BSP User Manual

stm32072b_eval.h File Reference

This file contains definitions for STM32072B_EVAL's Leds, push-buttons and COM port hardware resources. More...

```
#include "stm32f0xx_hal.h"
```

Go to the source code of this file.

Defines

#define	**LEDn** 4
#define	**LED1_PIN** GPIO_PIN_8
#define	**LED1_GPIO_PORT** GPIOD
#define	**LED1_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOD_CLK_E
#define	**LED1_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOD_CLK_D
#define	**LED2_PIN** GPIO_PIN_9
#define	**LED2_GPIO_PORT** GPIOD
#define	**LED2_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOD_CLK_E
#define	**LED2_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOD_CLK_D
#define	**LED3_PIN** GPIO_PIN_10
#define	**LED3_GPIO_PORT** GPIOD
#define	**LED3_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOD_CLK_E
#define	**LED3_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOD_CLK_D
#define	**LED4_PIN** GPIO_PIN_11
#define	**LED4_GPIO_PORT** GPIOD
#define	**LED4_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOD_CLK_E
#define	**LED4_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOD_CLK_D
#define	**LEDx_GPIO_CLK_ENABLE**(__LED__)
#define	**LEDx_GPIO_CLK_DISABLE**(__LED__)
#define	**JOYn** 5
#define	**BUTTONn** 1
#define	**TAMPER_BUTTON_PIN** GPIO_PIN_13 Tamper push-button.
#define	**TAMPER_BUTTON_GPIO_PORT** GPIOC
#define	**TAMPER_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_G
#define	**TAMPER_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_G
#define	**TAMPER_BUTTON_EXTI_IRQn** EXTI4_15_IRQn
#define	**TAMPERx_GPIO_CLK_ENABLE**(__BUTTON__) do { if((__B **BUTTON_TAMPER**) **TAMPER_BUTTON_GPIO_CLK_ENABL**
#define	**TAMPERx_GPIO_CLK_DISABLE**(__BUTTON__) (((__BUTT **BUTTON_TAMPER**) ? **TAMPER_BUTTON_GPIO_CLK_DISA**

#define **RIGHT_JOY_PIN** GPIO_PIN_3
Joystick Right push-button.

#define **RIGHT_JOY_GPIO_PORT** GPIOE

#define **RIGHT_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_(

#define **RIGHT_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_

#define **RIGHT_JOY_EXTI_IRQn** EXTI2_3_IRQn

#define **LEFT_JOY_PIN** GPIO_PIN_2
Joystick Left push-button.

#define **LEFT_JOY_GPIO_PORT** GPIOE

#define **LEFT_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_C

#define **LEFT_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_C

#define **LEFT_JOY_EXTI_IRQn** EXTI2_3_IRQn

#define **UP_JOY_PIN** GPIO_PIN_9
Joystick Up push-button.

#define **UP_JOY_GPIO_PORT** GPIOF

#define **UP_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOF_CLK

#define **UP_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOF_CLK

#define **UP_JOY_EXTI_IRQn** EXTI4_15_IRQn

#define **DOWN_JOY_PIN** GPIO_PIN_10
Joystick Down push-button.

#define **DOWN_JOY_GPIO_PORT** GPIOF

#define **DOWN_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOF_(

#define **DOWN_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOF_

#define **DOWN_JOY_EXTI_IRQn** EXTI4_15_IRQn

#define **SEL_JOY_PIN** GPIO_PIN_0
Joystick Sel push-button.

#define **SEL_JOY_GPIO_PORT** GPIOA

#define **SEL_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOA_CL

#define **SEL_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOA_CL

#define **SEL_JOY_EXTI_IRQn** EXTI0_1_IRQn

#define **JOYx_GPIO_CLK_ENABLE**(__JOY__)

#define **JOYx_GPIO_CLK_DISABLE**(__JOY__)

#define **COMn** 1

#define **EVAL_COM1** USART2

Definition for COM port1, connected to USART2.

```
#define EVAL_COM1_CLK_ENABLE()   __HAL_RCC_USART2_CLK_
#define EVAL_COM1_CLK_DISABLE()  __HAL_RCC_USART2_CLK
#define EVAL_COM1_TX_PIN         GPIO_PIN_5
#define EVAL_COM1_TX_GPIO_PORT   GPIOD
#define EVAL_COM1_TX_GPIO_CLK_ENABLE()   __HAL_RCC_GPI
#define EVAL_COM1_TX_GPIO_CLK_DISABLE()  __HAL_RCC_GP
#define EVAL_COM1_TX_AF          GPIO_AF0_USART2
#define EVAL_COM1_RX_PIN         GPIO_PIN_6
#define EVAL_COM1_RX_GPIO_PORT   GPIOD
#define EVAL_COM1_RX_GPIO_CLK_ENABLE()   __HAL_RCC_GPI
#define EVAL_COM1_RX_GPIO_CLK_DISABLE()  __HAL_RCC_GP
#define EVAL_COM1_RX_AF          GPIO_AF0_USART2
#define EVAL_COM1_CTS_PIN        GPIO_PIN_3
#define EVAL_COM1_CTS_GPIO_PORT  GPIOD
#define EVAL_COM1_CTS_GPIO_CLK_ENABLE()  __HAL_RCC_GI
#define EVAL_COM1_CTS_GPIO_CLK_DISABLE() __HAL_RCC_G
#define EVAL_COM1_CTS_AF         GPIO_AF0_USART2
#define EVAL_COM1_RTS_PIN        GPIO_PIN_4
#define EVAL_COM1_RTS_GPIO_PORT  GPIOD
#define EVAL_COM1_RTS_GPIO_CLK_ENABLE()  __HAL_RCC_GI
#define EVAL_COM1_RTS_GPIO_CLK_DISABLE() __HAL_RCC_G
#define EVAL_COM1_RTS_AF         GPIO_AF0_USART2
#define EVAL_COM1_IRQn           USART2_IRQn
#define COMx_CLK_ENABLE(__COM__)   do { if((__COM__) == COI
                                   EVAL_COM1_CLK_ENABLE();} while(0)
#define COMx_CLK_DISABLE(__COM__)  (((__COM__) == COM1) 
                                   EVAL_COM1_CLK_DISABLE() : 0)
#define COMx_TX_GPIO_CLK_ENABLE(__COM__)   do { if((__COM
                                   EVAL_COM1_TX_GPIO_CLK_ENABLE();} while(0)
#define COMx_TX_GPIO_CLK_DISABLE(__COM__)  (((__COM__) =
                                   EVAL_COM1_TX_GPIO_CLK_DISABLE() : 0)
#define COMx_RX_GPIO_CLK_ENABLE(__COM__)   do { if((__COM
                                   EVAL_COM1_RX_GPIO_CLK_ENABLE();} while(0)
```

```
#define COMx_RX_GPIO_CLK_DISABLE(__COM__)   (((__COM__) =
                EVAL_COM1_RX_GPIO_CLK_DISABLE() : 0)
#define COMx_CTS_GPIO_CLK_ENABLE(__COM__)   do { if((__COM
                EVAL_COM1_CTS_GPIO_CLK_ENABLE();} while(0)
#define COMx_CTS_GPIO_CLK_DISABLE(__COM__)   (((__COM__
                EVAL_COM1_CTS_GPIO_CLK_DISABLE() : 0)
#define COMx_RTS_GPIO_CLK_ENABLE(__COM__)   do { if((__COM
                EVAL_COM1_RTS_GPIO_CLK_ENABLE();} while(0)
#define COMx_RTS_GPIO_CLK_DISABLE(__COM__)   (((__COM__
                EVAL_COM1_RTS_GPIO_CLK_DISABLE() : 0)
#define EVAL_I2C1   I2C1
#define EVAL_I2C1_CLK_ENABLE()   __HAL_RCC_I2C1_CLK_ENAB
#define EVAL_I2C1_CLK_DISABLE()   __HAL_RCC_I2C1_CLK_DISA
#define EVAL_I2C1_FORCE_RESET()   __HAL_RCC_I2C1_FORCE_
#define EVAL_I2C1_RELEASE_RESET()   __HAL_RCC_I2C1_RELEA
#define EVAL_I2C1_SCL_PIN   GPIO_PIN_6 /* PB.6 */
#define EVAL_I2C1_SDA_PIN   GPIO_PIN_7 /* PB.7 */
#define EVAL_I2C1_GPIO_PORT   GPIOB /* GPIOB */
#define EVAL_I2C1_GPIO_CLK_ENABLE()   __HAL_RCC_GPIOB_C
#define EVAL_I2C1_GPIO_CLK_DISABLE()   __HAL_RCC_GPIOB_C
#define EVAL_I2C1_SCL_SDA_AF   GPIO_AF1_I2C1
#define EVAL_I2C2   I2C2
#define EVAL_I2C2_CLK_ENABLE()   __HAL_RCC_I2C2_CLK_ENAB
#define EVAL_I2C2_CLK_DISABLE()   __HAL_RCC_I2C2_CLK_DISA
#define EVAL_I2C2_FORCE_RESET()   __HAL_RCC_I2C2_FORCE_
#define EVAL_I2C2_RELEASE_RESET()   __HAL_RCC_I2C2_RELEA
#define EVAL_I2C2_SCL_PIN   GPIO_PIN_13 /* PB.13 */
#define EVAL_I2C2_SDA_PIN   GPIO_PIN_14 /* PB.14 */
#define EVAL_I2C2_GPIO_PORT   GPIOB /* GPIOB */
#define EVAL_I2C2_GPIO_CLK_ENABLE()   __HAL_RCC_GPIOB_C
#define EVAL_I2C2_GPIO_CLK_DISABLE()   __HAL_RCC_GPIOB_C
#define EVAL_I2C2_AF   GPIO_AF5_I2C2
#define EVAL_I2C2_IRQn   I2C2_IRQn
#define EVAL_I2C1_TIMEOUT_MAX   1000
#define EVAL_I2C2_TIMEOUT_MAX   1000
```

#define	**I2C2_TIMING** 0x00E0D3FF
#define	**I2C1_TIMING** 0x00E0D3FF
#define	**EVAL_SPIx** SPI1 Definition for SPI Interface pins (SPI1 used)
#define	**EVAL_SPIx_CLK_ENABLE**() __HAL_RCC_SPI1_CLK_ENA
#define	**EVAL_SPIx_CLK_DISABLE**() __HAL_RCC_SPI1_CLK_DIS
#define	**EVAL_SPIx_FORCE_RESET**() __HAL_RCC_SPI1_FORCE_
#define	**EVAL_SPIx_RELEASE_RESET**() __HAL_RCC_SPI1_RELE
#define	**EVAL_SPIx_SCK_PIN** GPIO_PIN_3 /* PB.03 */
#define	**EVAL_SPIx_SCK_GPIO_PORT** GPIOB /* GPIOB */
#define	**EVAL_SPIx_SCK_GPIO_CLK_ENABLE**() __HAL_RCC_GPI
#define	**EVAL_SPIx_SCK_GPIO_CLK_DISABLE**() __HAL_RCC_GP
#define	**EVAL_SPIx_SCK_AF** GPIO_AF0_SPI1
#define	**EVAL_SPIx_MISO_PIN** GPIO_PIN_14 /* PE.14 */
#define	**EVAL_SPIx_MISO_GPIO_PORT** GPIOE /* GPIOE */
#define	**EVAL_SPIx_MISO_GPIO_CLK_ENABLE**() __HAL_RCC_GP
#define	**EVAL_SPIx_MISO_GPIO_CLK_DISABLE**() __HAL_RCC_G
#define	**EVAL_SPIx_MISO_AF** GPIO_AF1_SPI1
#define	**EVAL_SPIx_MOSI_PIN** GPIO_PIN_15 /* PE.15 */
#define	**EVAL_SPIx_MOSI_GPIO_PORT** GPIOE /* GPIOE */
#define	**EVAL_SPIx_MOSI_GPIO_CLK_ENABLE**() __HAL_RCC_GP
#define	**EVAL_SPIx_MOSI_GPIO_CLK_DISABLE**() __HAL_RCC_G
#define	**EVAL_SPIx_MOSI_AF** GPIO_AF1_SPI1
#define	**EVAL_SPIx_MOSI_DIR_PIN** GPIO_PIN_2 /* PB.02 */
#define	**EVAL_SPIx_MOSI_DIR_GPIO_PORT** GPIOB /* GPIOB */
#define	**EVAL_SPIx_MOSI_DIR_GPIO_CLK_ENABLE**() __HAL_RC
#define	**EVAL_SPIx_MOSI_DIR_GPIO_CLK_DISABLE**() __HAL_RC
#define	**EVAL_SPIx_TIMEOUT_MAX** 1000
#define	**LCD_CS_LOW**() HAL_GPIO_WritePin(**LCD_NCS_GPIO_PO** GPIO_PIN_RESET)
#define	**LCD_CS_HIGH**() HAL_GPIO_WritePin(**LCD_NCS_GPIO_PC** GPIO_PIN_SET)
#define	**LCD_NCS_PIN** GPIO_PIN_6 /* PE. 06*/ LCD Control pins.

#define	**LCD_NCS_GPIO_PORT** GPIOE /* 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_2 /* PF.2 */ SD card Control pin.
#define	**SD_CS_GPIO_PORT** GPIOF /* GPIOF */
#define	**SD_CS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOF_CLK_
#define	**SD_CS_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOF_CLK_
#define	**SD_DETECT_PIN** GPIO_PIN_15 /* PB.15 */ SD Detect Interface pins.
#define	**SD_DETECT_GPIO_PORT** GPIOB /* GPIOB */
#define	**SD_DETECT_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_
#define	**SD_DETECT_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_
#define	**SD_DETECT_EXTI_IRQn** EXTI4_15_IRQn
#define	**HDMI_CEC_HPD_SINK_PIN** GPIO_PIN_15 /* PD.15 */ I2C HDMI CEC Interface pins.
#define	**HDMI_CEC_HPD_SINK_GPIO_PORT** GPIOD
#define	**HDMI_CEC_HPD_SINK_CLK_ENABLE**() __HAL_RCC_GPI
#define	**HDMI_CEC_HPD_SINK_CLK_DISABLE**() __HAL_RCC_GP
#define	**HDMI_CEC_HPD_SOURCE_PIN** GPIO_PIN_0 /* PE.0 */
#define	**HDMI_CEC_HPD_SOURCE_GPIO_PORT** GPIOE
#define	**HDMI_CEC_HPD_SOURCE_CLK_ENABLE**() __HAL_RCC_
#define	**HDMI_CEC_HPD_SOURCE_CLK_DISABLE**() __HAL_RCC_
#define	**HDMI_CEC_LINE_PIN** GPIO_PIN_8 /* PB.8 */
#define	**HDMI_CEC_LINE_GPIO_PORT** GPIOB
#define	**HDMI_CEC_LINE_CLK_ENABLE**() __HAL_RCC_GPIOB_CL
#define	**HDMI_CEC_LINE_CLK_DISABLE**() __HAL_RCC_GPIOB_C
#define	**HDMI_CEC_LINE_AF** GPIO_AF0_CEC
#define	**HDMI_CEC_IRQn** CEC_CAN_IRQn
#define	**HDMI_CEC_I2C_ADDRESS** 0xA0

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_TAMPER** = 0 } BUTTON Types Definition. More...
enum	**ButtonMode_TypeDef** { **BUTTON_MODE_GPIO** = 0, **BUTTON_MODE_EXTI** = 1 }

enum	**JOYState_TypeDef** {
	JOY_SEL = 0, **JOY_DOWN** = 1, **JOY_LEFT** = 2, **JOY_RIGHT** = 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 STM32F072B 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** Mode) Configures Tamper Button GPIO or EXTI Line.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
uint8_t	**BSP_JOY_Init** (**JOYMode_TypeDef** Joy_Mode) Configures joystick GPIO and EXTI modes.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the current joystick status.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) Configures COM port.

Detailed Description

This file contains definitions for STM32072B_EVAL's Leds, push-buttons and COM port hardware resources.

Author:

MCD Application Team

Attention:

Definition in file **stm32072b_eval.h**.

STM32072B_EVAL BSP User Manual

stm32072b_eval_eeprom.c File Reference

This file provides a set of functions needed to manage a M24LR64 I2C EEPROM memory. More...

```
#include "stm32072b_eval_eeprom.h"
```

Go to the source code of this file.

Functions

static uint32_t	**EEPROM_I2C_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
static uint32_t	**EEPROM_I2C_ReadBuffer** (uint8_t *pBuffer, uint16_t ReadAddr, uint32_t *NumByteToRead) Reads a block of data from the I2C EEPROM.
static uint32_t	**EEPROM_I2C_WritePage** (uint8_t *pBuffer, uint16_t WriteAddr, uint32_t *NumByteToWrite) Writes more than one byte to the EEPROM with a single WRITE cycle.
static uint32_t	**EEPROM_I2C_WaitEepromStandbyState** (void) Wait for EEPROM I2C Standby state.
uint32_t	**BSP_EEPROM_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
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**

Detailed Description

This file provides a set of functions needed to manage a M24LR64 I2C EEPROM memory.

Author:

MCD Application Team

==========

Notes:

- This driver is intended for STM32F0xx families devices only.
- The I2C EEPROM memory (M24LR64) is available on separate daughter board ANT7-M24LR-A, which is provided with the STM32072B EVAL board. To use this driver with M24LR64, you have to connect the ANT7-M24LR-A to CN2 connector of STM32072B EVAL board.

==========

It implements a high level communication layer for read and write from/to this memory. The needed STM32F0xx hardware resources (I2C and GPIO) are defined in **stm32072b_eval.h** file, and the initialization is performed depending of EEPROMs in **EEPROM_IO_Init()** function declared in **stm32072b_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **EEPROM_IO_Init()** function.

Note:

In this driver, basic read and write functions (**BSP_EEPROM_ReadBuffer()** and **BSP_EEPROM_WriteBuffer()**) use Polling mode to perform the data transfer to/from EEPROM memories.

+---+ | Pin assignment for M24LR64 EEPROM | +---------------------------------------+-----------+-------------+ | STM32F0xx I2C Pins | EEPROM | Pin | +---------------------------------------+-----------+-------------+ | EEPROM_I2C_SDA_PIN (PB7)/ SDA | SDA | 1 | | . | NC | 2 | | EEPROM_I2C_SCL_PIN/ SCL | SCL | 3 | | EX_RESET(PD7) | RESET | 4 | | . | VDD | 5 | | . | NC | 6 | | . | GND | 7 | |

. | NC | 8 | +----------+----------+----------+

Attention:

Definition in file **stm32072b_eval_eeprom.c**.

STM32072B_EVAL BSP User Manual

stm32072b_eval_eeprom.h File Reference

This file contains all the functions prototypes for the **stm32072b_eval_eeprom.c** firmware driver. More...

```
#include "stm32072b_eval.h"
```

Go to the source code of this file.

Data Structures

struct	**EEPROM_DrvTypeDef**

Defines

#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_M24LR64** 4 /* RF EEPROM ANT7-M24LR-A used */
#define	**EEPROM_OK** 0
#define	**EEPROM_FAIL** 1
#define	**EEPROM_TIMEOUT** 2
#define	**BSP_EEPROM_M24LR64** 1 /* RF I2C EEPROM M24LR64 */
#define	**EEPROM_MAX_TRIALS** 300

Functions

uint32_t	**BSP_EEPROM_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
uint32_t	**BSP_EEPROM_ReadBuffer** (uint8_t *pBuffer, uint16_t ReadAddr, 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_IO_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_IO_WriteData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Write data to I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_IO_ReadData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Read data from I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_IO_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.

Detailed Description

This file contains all the functions prototypes for the **stm32072b_eval_eeprom.c** firmware driver.

Author:

MCD Application Team

Attention:

Definition in file **stm32072b_eval_eeprom.h**.

STM32072B_EVAL BSP User Manual

stm32072b_eval_lcd.c File Reference

This file includes the driver for Liquid Crystal Display modules mounted on STM32072B-EVAL evaluation board. More...

```
#include "stm32072b_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 STM32072B-EVAL evaluation board.

Author:

MCD Application Team

Attention:

Definition in file **stm32072b_eval_lcd.c**.

STM32072B_EVAL BSP User Manual

Data Structures | Defines | Typedefs | Enumerations | Functions

stm32072b_eval_lcd.h File Reference

This file contains all the functions prototypes for the **stm32072b_eval_lcd.c** driver. More...

```
#include "stm32072b_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 **stm32072b_eval_lcd.c** driver.

Author:

MCD Application Team

Attention:

Definition in file **stm32072b_eval_lcd.h**.

STM32072B_EVAL BSP User Manual

Data Structures | Defines | Enumerations | Functions | Variables

stm32072b_eval_sd.c File Reference

This file provides a set of functions needed to manage the SPI SD Card memory mounted on STM32072B-EVAL board. It implements a high level communication layer for read and write from/to this memory. The needed STM32F0xx hardware resources (SPI and GPIO) are defined in **stm32072b_eval.h** file, and the initialization is performed in **SD_IO_Init()** function declared in **stm32072b_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **SD_IO_Init()** function. More...

```
#include "stm32072b_eval_sd.h" #include "stm32f0xx_hal.h"
#include "stdlib.h"
#include "string.h"
#include "stdio.h"
```

Go to the source code of this file.

Data Structures

struct	**SD_CmdAnswer_typedef**

Defines

#define	**SD_DUMMY_BYTE** 0xFF
#define	**SD_MAX_FRAME_LENGTH** 17 /* Lenght = 16 + 1 */
#define	**SD_CMD_LENGTH** 6
#define	**SD_MAX_TRY** 100 /* Number of try */
#define	**SD_CSD_STRUCT_V1** 0x2 /* CSD struct version V1 */
#define	**SD_CSD_STRUCT_V2** 0x1 /* CSD struct version V2 */
#define	**SD_TOKEN_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_TOKEN_START_DATA_MULTIPLE_BLOCK_READ** 0xF /* Data token start byte, Start Multiple Block Read */
#define	**SD_TOKEN_START_DATA_SINGLE_BLOCK_WRITE** 0xFE Data token start byte, Start Single Block Write */
#define	**SD_TOKEN_START_DATA_MULTIPLE_BLOCK_WRITE** 0x /* Data token start byte, Start Multiple Block Write */
#define	**SD_TOKEN_STOP_DATA_MULTIPLE_BLOCK_WRITE** 0xF /* Data toke stop byte, Stop Multiple Block Write */
#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_IF_COND** 8 /* CMD8 = 0x48 */
#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 SD_CMD_SD_APP_OP_COND 41 /* CMD41 = 0x69 */
#define SD_CMD_APP_CMD 55 /* CMD55 = 0x77 */
#define SD_CMD_READ_OCR 58 /* CMD55 = 0x79 */
```

Enumerations

enum	**SD_Answer_type** {
	SD_ANSWER_R1_EXPECTED,
	SD_ANSWER_R1B_EXPECTED,
	SD_ANSWER_R2_EXPECTED,
	SD_ANSWER_R3_EXPECTED,
	SD_ANSWER_R4R5_EXPECTED,
	SD_ANSWER_R7_EXPECTED
	}
	SD ansewer format. More...

enum	**SD_Error** {
	SD_R1_NO_ERROR = (0x00), **SD_R1_IN_IDLE_STATE** = (0x01), **SD_R1_ERASE_RESET** = (0x02),
	SD_R1_ILLEGAL_COMMAND = (0x04),
	SD_R1_COM_CRC_ERROR = (0x08),
	SD_R1_ERASE_SEQUENCE_ERROR = (0x10),
	SD_R1_ADDRESS_ERROR = (0x20),
	SD_R1_PARAMETER_ERROR = (0x40),
	SD_R2_NO_ERROR = 0x00, **SD_R2_CARD_LOCKED** = 0x01, **SD_R2_LOCKUNLOCK_ERROR** = 0x02,
	SD_R2_ERROR = 0x04,
	SD_R2_CC_ERROR = 0x08, **SD_R2_CARD_ECC_FAILED** = 0x10, **SD_R2_WP_VIOLATION** = 0x20,
	SD_R2_ERASE_PARAM = 0x40,
	SD_R2_OUTOFRANGE = 0x80, **SD_DATA_OK** = (0x05),
	SD_DATA_CRC_ERROR = (0x0B),
	SD_DATA_WRITE_ERROR = (0x0D),
	SD_DATA_OTHER_ERROR = (0xFF)
	}
	SD reponses and error flags. More...

Functions

static uint8_t	**SD_GetCIDRegister** (**SD_CID** *Cid) Reads the SD card CID register.
static uint8_t	**SD_GetCSDRegister** (**SD_CSD** *Csd) Reads the SD card SCD register.
static uint8_t	**SD_GetDataResponse** (void) Gets the SD card data response and check the busy flag.
static uint8_t	**SD_GoIdleState** (void) Put the SD in Idle state.
static **SD_CmdAnswer_typedef**	**SD_SendCmd** (uint8_t Cmd, uint32_t Arg, uint8_t Crc, uint8_t Answer) Send 5 bytes command to the SD card and get response.
static uint8_t	**SD_WaitData** (uint8_t data) Waits a data from the SD card.
static uint8_t	**SD_ReadData** (void) Waits a data until a value different from SD_DUMMY_BITE.
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 *pData, uint32_t ReadAddr, uint16_t BlockSize, uint32_t NumberOfBlocks) Reads block(s) from a specified address in the SD card, in polling mode.
uint8_t	**BSP_SD_WriteBlocks** (uint32_t *pData, uint32_t WriteAddr, uint16_t BlockSize, uint32_t NumberOfBlocks) Writes block(s) to a specified address in the 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.

Variables

__IO uint8_t	**SdStatus** = **SD_NOT_PRESENT**
uint16_t	**flag_SDHC** = 0

Detailed Description

This file provides a set of functions needed to manage the SPI SD Card memory mounted on STM32072B-EVAL board. It implements a high level communication layer for read and write from/to this memory. The needed STM32F0xx hardware resources (SPI and GPIO) are defined in **stm32072b_eval.h** file, and the initialization is performed in **SD_IO_Init()** function declared in **stm32072b_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **SD_IO_Init()** function.

Author:

MCD Application Team +----------+ | Pin assignment | +----------+----------+----------+ | STM32F0xx 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:

```
  ============================================================================
                      ##### How to use this driver #####
  ============================================================================
  [..]
```

```
   (#) This driver is used to drive the micro SD external card mounted on STM32072B-EVAL
       evaluation board.
   (#) This driver does not need a specific component driver for the micro SD device
       to be included with.

   (#) Initialization steps:
       (++) Initialize the micro SD card using the SD_Init() function.
       (++) To check the SD card presence you can use the function BSP_SD_IsDetected() which
            returns the detection status
       (++) The function BSP_SD_GetCardInfo() is used to get the micro SD card information
            which is stored in the structure "SD_CardInfo".

   (#) Micro SD card operations
       (++) The micro SD card can be accessed with read/write block(s) operations once
            it is reay for access. The access cand be performed in polling
            mode by calling the functions SD_ReadBlocks()/SD_WriteBlocks()
       (++) The SD erase block(s) is performed using the function BSP_SD_Erase() with
            specifying the number of blocks to erase.
       (++) The SD runtime status is returned when calling the function BSP_SD_GetStatus().
```

Definition in file **stm32072b_eval_sd.c**.

User Manual by  1.7.6.1

STM32072B_EVAL BSP User Manual

Data Structures | Defines | Enumerations | Functions

stm32072b_eval_sd.h File Reference

This file contains the common defines and functions prototypes for the **stm32072b_eval_sd.c** driver. More...

```
#include "stm32072b_eval.h"
```

Go to the source code of this file.

Data Structures

struct	**struct_v1**
struct	**struct_v2**
struct	**SD_CSD** Card Specific Data: CSD Register. More...
union	**SD_CSD::csd_version**
struct	**SD_CID** Card Identification Data: CID Register. More...
struct	**SD_CardInfo** SD Card information. More...

Defines

#define	**SD_BLOCK_SIZE** 0x200 Block Size.
#define	**SD_PRESENT** ((uint8_t)0x01) SD detection on its memory slot.
#define	**SD_NOT_PRESENT** ((uint8_t)0x00)

Enumerations

enum	{ **BSP_SD_OK** = 0x00, **MSD_OK** = 0x00, **BSP_SD_ERROR** = 0x01, **MSD_ERROR** = 0x01, **BSP_SD_TIMEOUT** } SD status structure definition. More...

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 *pData, uint32_t ReadAddr, uint16_t BlockSize, uint32_t NumberOfBlocks) Reads block(s) from a specified address in the SD card, in polling mode.
uint8_t	**BSP_SD_WriteBlocks** (uint32_t *pData, uint32_t WriteAddr, uint16_t BlockSize, uint32_t NumberOfBlocks) Writes block(s) to a specified address in the 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_CSState** (uint8_t state)
void	**SD_IO_WriteReadData** (const uint8_t *DataIn, uint8_t *DataOut, uint16_t DataLength) Write a byte on the SD.
uint8_t	**SD_IO_WriteByte** (uint8_t Data) Writes a byte on the SD.

Detailed Description

This file contains the common defines and functions prototypes for the **stm32072b_eval_sd.c** driver.

Author:

MCD Application Team

Attention:

Definition in file **stm32072b_eval_sd.h**.

STM32072B_EVAL BSP User Manual

stm32072b_eval_tsensor.c File Reference

This file provides a set of functions needed to manage the I2C STLM75 temperature sensor mounted on STM32072B-EVAL board . It implements a high level communication layer for read and write from/to this sensor. The needed STM32F0xx hardware resources (I2C and GPIO) are defined in **stm32072b_eval.h** file, and the initialization is performed in **TSENSOR_IO_Init()** function declared in **stm32072b_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 "stm32072b_eval_tsensor.h"
```

Go to the source code of this file.

Functions

uint32_t	**BSP_TSENSOR_Init** (void) Initializes peripherals used by the I2C Temperature Sensor driver.
uint8_t	**BSP_TSENSOR_ReadStatus** (void) Returns the Temperature Sensor status.
uint16_t	**BSP_TSENSOR_ReadTemp** (void) Read Temperature register of STLM75.

Variables

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

Detailed Description

This file provides a set of functions needed to manage the I2C STLM75 temperature sensor mounted on STM32072B-EVAL board . It implements a high level communication layer for read and write from/to this sensor. The needed STM32F0xx hardware resources (I2C and GPIO) are defined in **stm32072b_eval.h** file, and the initialization is performed in **TSENSOR_IO_Init()** function declared in **stm32072b_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 | +-------------------------------------+-------------+-------------+ | STM32F0xx I2C Pins | STLM75 | Pin | +-------------------------------------+-------------+-------------+ | STLM75_I2C_SDA_PIN/ SDA | SDA | 1 | | STLM75_I2C_SCL_PIN/ SCL | SCL | 2 | | STLM75_I2C_SMBUSALERT_PIN/ SMBUS ALERT| OS/INT | 3 | | . | GND | 4 (0V) | | . | GND | 5 (0V) | | . | GND | 6 (0V) | | . | GND | 7 (0V) | | . | VDD | 8 (3.3V)| +-------------------------------------+-------------+-------------+

Attention:

Definition in file **stm32072b_eval_tsensor.c**.

STM32072B_EVAL BSP User Manual

stm32072b_eval_tsensor.h File Reference

This file contains all the functions prototypes for the **stm32072b_eval_tsensor.c** firmware driver. More...

```
#include "stm32072b_eval.h" #include
"../Components/stlm75/stlm75.h"
```

Go to the source code of this file.

Defines

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

Enumerations

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

Functions

uint32_t	**BSP_TSENSOR_Init** (void) Initializes peripherals used by the I2C Temperature Sensor driver.
uint8_t	**BSP_TSENSOR_ReadStatus** (void) Returns the Temperature Sensor status.
uint16_t	**BSP_TSENSOR_ReadTemp** (void) Read Temperature register of STLM75.

Detailed Description

This file contains all the functions prototypes for the **stm32072b_eval_tsensor.c** firmware driver.

Author:
MCD Application Team

Attention:

Definition in file **stm32072b_eval_tsensor.h**.

STM32072B_EVAL BSP User Manual

Modules

Here is a list of all modules:

STM32072B_EVAL BSP User Manual

Data Structures

Here are the data structures with brief descriptions:

SD_CSD::csd_version	
EEPROM_DrvTypeDef	
LCD_DrawPropTypeDef	
Point	
SD_CardInfo	SD Card information
SD_CID	Card Identification Data: CID Register
SD_CmdAnswer_typedef	
SD_CSD	Card Specific Data: CSD Register
struct_v1	
struct_v2	

STM32072B_EVAL BSP User Manual

File List

Here is a list of all files with brief descriptions:

stm32072b_eval.c [code]	This file provides: a set of firmware functions to manage Leds, push-button and COM ports
stm32072b_eval.h [code]	This file contains definitions for STM32072B_EVAL's Leds, push-buttons and COM port hardware resources
stm32072b_eval_eeprom.c [code]	This file provides a set of functions needed to manage a M24LR64 I2C EEPROM memory
stm32072b_eval_eeprom.h [code]	This file contains all the functions prototypes for the **stm32072b_eval_eeprom.c** firmware driver
stm32072b_eval_lcd.c [code]	This file includes the driver for Liquid Crystal Display modules mounted on STM32072B-EVAL evaluation board
stm32072b_eval_lcd.h [code]	This file contains all the

	functions prototypes for the **stm32072b_eval_lcd.c** driver
stm32072b_eval_sd.c [code]	This file provides a set of functions needed to manage the SPI SD Card memory mounted on STM32072B-EVAL board. It implements a high level communication layer for read and write from/to this memory. The needed STM32F0xx hardware resources (SPI and GPIO) are defined in **stm32072b_eval.h** file, and the initialization is performed in **SD_IO_Init()** function declared in **stm32072b_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **SD_IO_Init()** function
stm32072b_eval_sd.h [code]	This file contains the common defines and functions prototypes for the **stm32072b_eval_sd.c** driver
stm32072b_eval_tsensor.c [code]	This file provides a set of functions needed to manage the I2C STLM75 temperature sensor mounted on STM32072B-EVAL board . It implements a high level communication layer for read and write from/to this sensor. The needed STM32F0xx hardware resources (I2C and

	GPIO) are defined in **stm32072b_eval.h** file, and the initialization is performed in **TSENSOR_IO_Init()** function declared in **stm32072b_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
stm32072b_eval_tsensor.h [code]	This file contains all the functions prototypes for the **stm32072b_eval_tsensor.c** firmware driver

STM32072B_EVAL BSP User Manual

Directories

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

- Firmware
  - Drivers
    - BSP
      - STM32072B_EVAL

STM32072B_EVAL BSP User Manual

Data Structure Index

C | E | L | P | S

C | E | L | P | S

STM32072B_EVAL BSP User Manual

Exported Types

STM32072B_EVAL EEPROM

Data Structures

struct	**EEPROM_DrvTypeDef**

STM32072B_EVAL BSP User Manual

stm32072b_eval_eeprom.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32072b_eval_eeprom.h
00004   * @author  MCD Application Team
00005   * @brief   This file contains all the functions prototypes for
00006   *          the stm32072b_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 __STM32072B_EVAL_EEPROM_H
00039 #define __STM32072B_EVAL_EEPROM_H
00040
00041 #ifdef __cplusplus
00042  extern "C" {
00043 #endif
00044
00045 /* Includes ------------------------------
-----------------------------------*/
00046 #include "stm32072b_eval.h"
00047
00048 /** @addtogroup BSP
00049   * @{
00050   */
00051
00052 /** @addtogroup STM32072B_EVAL
00053   * @{
00054   */
00055
00056 /** @defgroup STM32072B_EVAL_EEPROM STM32072
B_EVAL EEPROM
00057   * @{
00058   */
00059
00060 /** @defgroup STM32072B_EVAL_EEPROM_Exported
_Types Exported Types
00061   * @{
00062   */
00063 typedef struct
00064 {
00065   uint32_t  (*Init)(void);
00066   uint32_t  (*ReadBuffer)(uint8_t* , uint16_
```

```
t , uint32_t* );
00067   uint32_t  (*WritePage)(uint8_t* , uint16_t
 , uint32_t* );
00068 }EEPROM_DrvTypeDef;
00069 /**
00070   * @}
00071   */
00072
00073 /** @defgroup STM32072B_EVAL_EEPROM_Exported
_Constants Exported Constants
00074   * @{
00075   */
00076 /* EEPROMs hardware address and page size */

00077 #define EEPROM_ADDRESS_M24LR64_A01      0xA0
    /* RF EEPROM ANT7-M24LR-A01 used */
00078 #define EEPROM_ADDRESS_M24LR64_A02      0xA6
    /* RF EEPROM ANT7-M24LR-A02 used */
00079
00080 #define EEPROM_PAGESIZE_M24LR64         4
    /* RF EEPROM ANT7-M24LR-A used */
00081
00082 /* EEPROM BSP return values */
00083 #define EEPROM_OK                       0
00084 #define EEPROM_FAIL                     1
00085 #define EEPROM_TIMEOUT                  2
00086
00087 /* EEPROM BSP devices definition list suppor
ted */
00088 #define BSP_EEPROM_M24LR64              1
    /* RF I2C EEPROM M24LR64 */
00089
00090 /* Maximum number of trials for EEPROM_I2C_W
aitEepromStandbyState() function */
00091 #define EEPROM_MAX_TRIALS               300
00092 /**
00093   * @}
```

```
00094   */
00095
00096 /** @defgroup STM32072B_EVAL_EEPROM_Exported
_Functions Exported Functions
00097   * @{
00098   */
00099 uint32_t          BSP_EEPROM_Init(void);
00100 uint32_t          BSP_EEPROM_ReadBuffer(uint
8_t* pBuffer, uint16_t ReadAddr, uint32_t* NumByte
ToRead);
00101 uint32_t          BSP_EEPROM_WriteBuffer(uin
t8_t* pBuffer, uint16_t WriteAddr, uint32_t NumByt
eToWrite);
00102
00103 /* USER Callbacks: This function is declared
 as __weak in EEPROM driver and
00104    should be implemented into user applicati
on.
00105    BSP_EEPROM_TIMEOUT_UserCallback() functio
n is called whenever a timeout condition
00106    occure during communication (waiting on a
n event that doesn't occur, bus
00107    errors, busy devices ...). */
00108 void              BSP_EEPROM_TIMEOUT_UserCal
lback(void);
00109
00110 /**
00111   * @}
00112   */
00113
00114 /** @defgroup STM32072B_EVAL_EEPROM_LINK_Ope
rations_Functions LINK Operations Functions
00115   * @{
00116   */
00117
00118 /* Link functions for I2C EEPROM peripheral
*/
```

```
00119 void              EEPROM_IO_Init(void);
00120 HAL_StatusTypeDef EEPROM_IO_WriteData(uint16
_t DevAddress, uint16_t MemAddress, uint8_t* pBuff
er, uint32_t BufferSize);
00121 HAL_StatusTypeDef EEPROM_IO_ReadData(uint16_
t DevAddress, uint16_t MemAddress, uint8_t* pBuffe
r, uint32_t BufferSize);
00122 HAL_StatusTypeDef EEPROM_IO_IsDeviceReady(ui
nt16_t DevAddress, uint32_t Trials);
00123
00124 /**
00125   * @}
00126   */
00127
00128 /**
00129   * @}
00130   */
00131
00132 /**
00133   * @}
00134   */
00135
00136 /**
00137   * @}
00138   */
00139
00140 #ifdef __cplusplus
00141 }
00142 #endif
00143
00144 #endif /* __STM32072B_EVAL_EEPROM_H */
00145
00146 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32072B_EVAL BSP User Manual

stm32072b_eval_eeprom.c

Go to the documentation of this file.

```
00001 /**
00002 ******************************************************************************
00003 * @file    stm32072b_eval_eeprom.c
00004 * @author  MCD Application Team
00005 * @brief   This file provides a set of functions needed to manage a M24LR64
00006 *          I2C EEPROM memory.
00007 *
00008 *          ===================================
00009 *          Notes:
00010 *           - This driver is intended for STM32F0xx families devices only.
00011 *           - The I2C EEPROM memory (M24LR64) is available on separate daughter
00012 *             board ANT7-M24LR-A, which is provided with the STM32072B
00013 *             EVAL board.
00014 *             To use this driver with M24LR64, you have to connect
00015 *             the ANT7-M24LR-A to CN2 connec
```

```
tor of STM32072B EVAL board.
00016 *          ================================
================================
00017 *
00018 *          It implements a high level commun
ication layer for read and write
00019 *          from/to this memory. The needed S
TM32F0xx hardware resources (I2C
00020 *          and GPIO) are defined in stm32072
b_eval.h file,
00021 *          and the initialization is perform
ed depending of EEPROMs
00022 *          in EEPROM_IO_Init() function decl
ared in stm32072b_eval.c file.
00023 *          You can easily tailor this driver
 to any other development board,
00024 *          by just adapting the defines for
hardware resources and
00025 *          EEPROM_IO_Init() function.
00026 *
00027 *          @note In this driver, basic read
and write functions
00028 *          (BSP_EEPROM_ReadBuffer() and BSP_
EEPROM_WriteBuffer())
00029 *          use Polling mode to perform the d
ata transfer to/from EEPROM memories.
00030
00031 *     +-----------------------------------
----------------------------+
00032 *     |                Pin assignment for M24
LR64 EEPROM                 |
00033 *     +-----------------------------------
--+-----------+-------------+
00034 *     |  STM32F0xx I2C Pins
  |   EEPROM  |   Pin       |
00035 *     +-----------------------------------
--+-----------+-------------+
```

```
00036 *     | EEPROM_I2C_SDA_PIN (PB7)/ SDA
  |   SDA     |    1         |
00037 *     | .
  |   NC      |    2         |
00038 *     | EEPROM_I2C_SCL_PIN/ SCL
  |   SCL     |    3         |
00039 *     | EX_RESET(PD7)
  |   RESET   |    4         |
00040 *     | .
  |   VDD     |    5         |
00041 *     | .
  |   NC      |    6         |
00042 *     | .
  |   GND     |    7         |
00043 *     | .
  |   NC      |    8         |
00044 *     +-----------------------------------
--+-----------+-------------+
00045 *
00046 ******************************************
*********************************
00047 * @attention
00048 *
00049 * <h2><center>&copy; COPYRIGHT(c) 2016 STMic
roelectronics</center></h2>
00050 *
00051 * Redistribution and use in source and binar
y forms, with or without modification,
00052 * are permitted provided that the following
conditions are met:
00053 *   1. Redistributions of source code must r
etain the above copyright notice,
00054 *      this list of conditions and the follo
wing disclaimer.
00055 *   2. Redistributions in binary form must r
eproduce the above copyright notice,
00056 *      this list of conditions and the follo
```

```
wing disclaimer in the documentation
00057 *      and/or other materials provided with
the distribution.
00058 *   3. Neither the name of STMicroelectronic
s nor the names of its contributors
00059 *      may be used to endorse or promote pro
ducts derived from this software
00060 *      without specific prior written permis
sion.
00061 *
00062 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
 HOLDERS AND CONTRIBUTORS "AS IS"
00063 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INC
LUDING, BUT NOT LIMITED TO, THE
00064 * IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS FOR A PARTICULAR PURPOSE ARE
00065 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGH
T HOLDER OR CONTRIBUTORS BE LIABLE
00066 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPEC
IAL, EXEMPLARY, OR CONSEQUENTIAL
00067 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PR
OCUREMENT OF SUBSTITUTE GOODS OR
00068 * SERVICES; LOSS OF USE, DATA, OR PROFITS; O
R BUSINESS INTERRUPTION) HOWEVER
00069 * CAUSED AND ON ANY THEORY OF LIABILITY, WHE
THER IN CONTRACT, STRICT LIABILITY,
00070 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE
) ARISING IN ANY WAY OUT OF THE USE
00071 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE P
OSSIBILITY OF SUCH DAMAGE.
00072 *
00073 ********************************************
**********************************
00074 */
00075
00076 /* Includes ------------------------------
----------------------------------*/
```

```
00077 #include "stm32072b_eval_eeprom.h"
00078
00079 /** @addtogroup BSP
00080 * @{
00081 */
00082
00083 /** @addtogroup STM32072B_EVAL
00084   * @{
00085   */
00086
00087 /** @addtogroup STM32072B_EVAL_EEPROM
00088   * @brief      This file includes the I2C E
EPROM driver
00089   *             of STM32072B-EVAL board.
00090   * @{
00091   */
00092
00093 /** @defgroup STM32072B_EVAL_EEPROM_Private_
Variables Private Variables
00094   * @{
00095   */
00096 __IO uint16_t  EEPROMAddress = 0;
00097 __IO uint16_t  EEPROMPageSize = 0;
00098 __IO uint16_t  EEPROMDataRead;
00099 __IO uint8_t   EEPROMDataWrite;
00100
00101 static EEPROM_DrvTypeDef *EEPROM_SelectedDev
ice = 0;
00102 /**
00103   * @}
00104   */
00105
00106 /** @defgroup STM32072B_EVAL_EEPROM_Private_
Functions Private Functions
00107   * @{
00108   */
00109 static uint32_t EEPROM_I2C_Init(void);
```

```
00110 static uint32_t EEPROM_I2C_ReadBuffer(uint8_
t* pBuffer, uint16_t ReadAddr, uint32_t* NumByteTo
Read);
00111 static uint32_t EEPROM_I2C_WritePage(uint8_t
* pBuffer, uint16_t WriteAddr, uint32_t* NumByteTo
Write);
00112 static uint32_t EEPROM_I2C_WaitEepromStandby
State(void);
00113
00114 /**
00115   * @}
00116   */
00117
00118 /** @defgroup STM32072B_EVAL_EEPROM_Private_
Types Private Types
00119   * @{
00120   */
00121 /* EEPROM I2C driver typedef */
00122 EEPROM_DrvTypeDef EEPROM_I2C_Drv =
00123 {
00124   EEPROM_I2C_Init,
00125   EEPROM_I2C_ReadBuffer,
00126   EEPROM_I2C_WritePage
00127 };
00128 /**
00129   * @}
00130   */
00131
00132 /** @addtogroup STM32072B_EVAL_EEPROM_Export
ed_Functions
00133   * @{
00134   */
00135
00136 /**
00137   * @brief  Initializes peripherals used by
the I2C EEPROM driver.
00138   *
```

```
00139   * @note There are 2 different versions of
M24LR64 (A01 & A02).
00140   *             Then try to connect on 1st o
ne (EEPROM_I2C_ADDRESS_A01)
00141   *             and if problem, check the 2n
d one (EEPROM_I2C_ADDRESS_A02)
00142   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00143   *         different from EEPROM_OK (0)
00144   */
00145 uint32_t BSP_EEPROM_Init(void)
00146 {
00147   EEPROM_SelectedDevice = &EEPROM_I2C_Drv;
00148   if(EEPROM_SelectedDevice->Init != 0)
00149   {
00150     return (EEPROM_SelectedDevice->Init());
00151   }
00152   else
00153   {
00154     return EEPROM_FAIL;
00155   }
00156 }
00157
00158 /**
00159   * @brief  Reads a block of data from the E
EPROM device selected.
00160   * @param  pBuffer  pointer to the buffer t
hat receives the data read from
00161   *         the EEPROM.
00162   * @param  ReadAddr  EEPROM's internal addr
ess to start reading from.
00163   * @param  NumByteToRead  pointer to the va
riable holding number of bytes to
00164   *         be read from the EEPROM.
00165   *
00166   *        @note The variable pointed by Num
ByteToRead is reset to 0 when all the
```

```
00167   *               data are read from the EEPR
OM. Application should monitor this
00168   *               variable in order know when
 the transfer is complete.
00169   *
00170   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00171   *         different from EEPROM_OK (0) or
the timeout user callback.
00172   */
00173 uint32_t BSP_EEPROM_ReadBuffer(uint8_t* pBuf
fer, uint16_t ReadAddr, uint32_t* NumByteToRead)
00174 {
00175   if(EEPROM_SelectedDevice->ReadBuffer != 0)
00176   {
00177     return (EEPROM_SelectedDevice->ReadBuffer
(pBuffer, ReadAddr, NumByteToRead));
00178   }
00179   else
00180   {
00181     return EEPROM_FAIL;
00182   }
00183 }
00184
00185 /**
00186   * @brief  Writes buffer of data to the EEP
ROM device selected.
00187   * @param  pBuffer  pointer to the buffer
containing the data to be written
00188   *         to the EEPROM.
00189   * @param  WriteAddr  EEPROM's internal add
ress to write to.
00190   * @param  NumByteToWrite  number of bytes
to write to the EEPROM.
00191   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00192   *         different from EEPROM_OK (0) or
```

```
the timeout user callback.
00193   */
00194 uint32_t BSP_EEPROM_WriteBuffer(uint8_t* pBu
ffer, uint16_t WriteAddr, uint32_t NumByteToWrite)
00195 {
00196   uint16_t numofpage = 0, numofsingle = 0, c
ount = 0;
00197   uint16_t addr = 0;
00198   uint32_t dataindex = 0;
00199   uint32_t status = EEPROM_OK;
00200
00201   addr = WriteAddr % EEPROMPageSize;
00202   count = EEPROMPageSize - addr;
00203   numofpage =  NumByteToWrite / EEPROMPageSi
ze;
00204   numofsingle = NumByteToWrite % EEPROMPageS
ize;
00205
00206   if(EEPROM_SelectedDevice->WritePage == 0)
00207   {
00208     return EEPROM_FAIL;
00209   }
00210
00211   /*!< If WriteAddr is EEPROM_PAGESIZE align
ed  */
00212   if(addr == 0)
00213   {
00214     /*!< If NumByteToWrite < EEPROM_PAGESIZE
 */
00215     if(numofpage == 0)
00216     {
00217       /* Store the number of data to be writ
ten */
00218       dataindex = numofsingle;
00219       /* Start writing data */
00220       status = EEPROM_SelectedDevice->WriteP
age(pBuffer, WriteAddr, (uint32_t*)(&dataindex));
```

```
00221       if (status != EEPROM_OK)
00222       {
00223         return status;
00224       }
00225     }
00226     /*!< If NumByteToWrite > EEPROM_PAGESIZE
 */
00227     else
00228     {
00229       while(numofpage--)
00230       {
00231         /* Store the number of data to be wr
itten */
00232         dataindex = EEPROMPageSize;
00233         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00234         if (status != EEPROM_OK)
00235         {
00236           return status;
00237         }
00238
00239         WriteAddr +=  EEPROMPageSize;
00240         pBuffer += EEPROMPageSize;
00241       }
00242
00243       if(numofsingle!=0)
00244       {
00245         /* Store the number of data to be wr
itten */
00246         dataindex = numofsingle;
00247         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00248         if (status != EEPROM_OK)
00249         {
00250           return status;
```

```
00251         }
00252       }
00253     }
00254   }
00255   /*!< If WriteAddr is not EEPROM_PAGESIZE a
ligned  */
00256   else
00257   {
00258     /*!< If NumByteToWrite < EEPROM_PAGESIZE
 */
00259     if(numofpage== 0)
00260     {
00261       /*!< If the number of data to be writt
en is more than the remaining space
00262       in the current page: */
00263       if (NumByteToWrite > count)
00264       {
00265         /* Store the number of data to be wr
itten */
00266         dataindex = count;
00267         /*!< Write the data contained in sam
e page */
00268         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00269         if (status != EEPROM_OK)
00270         {
00271           return status;
00272         }
00273
00274         /* Store the number of data to be wr
itten */
00275         dataindex = (NumByteToWrite - count)
;
00276         /*!< Write the remaining data in the
 following page */
00277         status = EEPROM_SelectedDevice->Writ
```

```
ePage((uint8_t*)(pBuffer + count), (WriteAddr + co
unt), (uint32_t*)(&dataindex));
00278         if (status != EEPROM_OK)
00279         {
00280           return status;
00281         }
00282       }
00283       else
00284       {
00285         /* Store the number of data to be wr
itten */
00286         dataindex = numofsingle;
00287         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00288         if (status != EEPROM_OK)
00289         {
00290           return status;
00291         }
00292       }
00293     }
00294     /*!< If NumByteToWrite > EEPROM_PAGESIZE
 */
00295     else
00296     {
00297       NumByteToWrite -= count;
00298       numofpage =  NumByteToWrite / EEPROMPa
geSize;
00299       numofsingle = NumByteToWrite % EEPROMP
ageSize;
00300
00301       if(count != 0)
00302       {
00303         /* Store the number of data to be wr
itten */
00304         dataindex = count;
00305         status = EEPROM_SelectedDevice->Writ
```

```
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00306         if (status != EEPROM_OK)
00307         {
00308           return status;
00309         }
00310         WriteAddr += count;
00311         pBuffer += count;
00312       }
00313
00314       while(numofpage--)
00315       {
00316         /* Store the number of data to be wr
itten */
00317         dataindex = EEPROMPageSize;

00318         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00319         if (status != EEPROM_OK)
00320         {
00321           return status;
00322         }
00323         WriteAddr +=  EEPROMPageSize;
00324         pBuffer += EEPROMPageSize;
00325       }
00326       if(numofsingle != 0)
00327       {
00328         /* Store the number of data to be wr
itten */
00329         dataindex = numofsingle;
00330         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00331         if (status != EEPROM_OK)
00332         {
00333           return status;
```

```
00334         }
00335       }
00336     }
00337   }
00338
00339   /* If all operations OK, return EEPROM_OK
(0) */
00340   return EEPROM_OK;
00341 }
00342
00343 /**
00344   * @brief  Basic management of the timeout
situation.
00345   * @retval None.
00346   */
00347 __weak void BSP_EEPROM_TIMEOUT_UserCallback(
void)
00348 {
00349 }
00350 /**
00351   * @}
00352   */
00353
00354 /** @addtogroup STM32072B_EVAL_EEPROM_Privat
e_Functions
00355   * @{
00356   */
00357
00358 /**
00359   * @brief  Initializes peripherals used by
the I2C EEPROM driver.
00360   *
00361   * @note There are 2 different versions of
M24LR64 (A01 & A02).
00362   *             Then try to connect on 1st o
ne (EEPROM_I2C_ADDRESS_A01)
00363   *             and if problem, check the 2n
```

```
d one (EEPROM_I2C_ADDRESS_A02)
00364   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00365   *         different from EEPROM_OK (0)
00366   */
00367 static uint32_t EEPROM_I2C_Init(void)
00368 {
00369   EEPROM_IO_Init();
00370
00371   /*Select the EEPROM address for M24LR64 A0
2 and check if OK*/
00372   EEPROMAddress = EEPROM_ADDRESS_M24LR64_A01
;
00373   EEPROMPageSize = EEPROM_PAGESIZE_M24LR64;
00374   if (EEPROM_IO_IsDeviceReady(EEPROMAddress,
EEPROM_MAX_TRIALS) != HAL_OK)
00375   {
00376     /*Select the EEPROM address for M24LR64
A01 and check if OK*/
00377     EEPROMAddress = EEPROM_ADDRESS_M24LR64_A
02;
00378     EEPROMPageSize = EEPROM_PAGESIZE_M24LR64
;
00379     if (EEPROM_IO_IsDeviceReady(EEPROMAddress
, EEPROM_MAX_TRIALS) != HAL_OK)
00380     {
00381       return EEPROM_FAIL;
00382     }
00383   }
00384
00385   return EEPROM_OK;
00386 }
00387
00388 /**
00389   * @brief  Reads a block of data from the I
2C EEPROM.
00390   * @param  pBuffer  pointer to the buffer t
```

```
hat receives the data read from
00391   *         the EEPROM.
00392   * @param  ReadAddr  EEPROM's internal addr
ess to start reading from.
00393   * @param  NumByteToRead  pointer to the va
riable holding number of bytes to
00394   *         be read from the EEPROM.
00395   *
00396   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00397   *         different from EEPROM_OK (0) or
the timeout user callback.
00398   */
00399 static uint32_t EEPROM_I2C_ReadBuffer(uint8_
t* pBuffer, uint16_t ReadAddr, uint32_t* NumByteTo
Read)
00400 {
00401   uint32_t buffersize = *NumByteToRead;
00402
00403   if (EEPROM_IO_ReadData(EEPROMAddress, Read
Addr, pBuffer, buffersize) != HAL_OK)
00404   {
00405     return EEPROM_FAIL;
00406   }
00407
00408   /* If all operations OK, return EEPROM_OK
(0) */
00409   return EEPROM_OK;
00410 }
00411
00412 /**
00413   * @brief  Writes more than one byte to the
 EEPROM with a single WRITE cycle.
00414   *
00415   * @note   The number of bytes (combined to
 write start address) must not
00416   *         cross the EEPROM page boundary.
```

```
This function can only write into
00417   *         the boundaries of an EEPROM page.

00418   *         This function doesn't check on b
oundaries condition (in this driver
00419   *         the function BSP_EEPROM_WriteBuf
fer() which calls EEPROM_WritePage() is
00420   *         responsible of checking on Page
boundaries).
00421   *
00422   * @param  pBuffer  pointer to the buffer c
ontaining the data to be written to
00423   *         the EEPROM.
00424   * @param  WriteAddr  EEPROM's internal add
ress to write to.
00425   * @param  NumByteToWrite  pointer to the v
ariable holding number of bytes to
00426   *         be written into the EEPROM.
00427   *
00428   *        @note The variable pointed by Num
ByteToWrite is reset to 0 when all the
00429   *              data are written to the EEP
ROM. Application should monitor this
00430   *              variable in order know when
 the transfer is complete.
00431   *
00432   *
00433   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00434   *         different from EEPROM_OK (0) or
the timeout user callback.
00435   */
00436 static uint32_t EEPROM_I2C_WritePage(uint8_t
* pBuffer, uint16_t WriteAddr, uint32_t* NumByteTo
Write)
00437 {
00438   uint32_t buffersize = *NumByteToWrite;
```

```
00439
00440   if (EEPROM_IO_WriteData(EEPROMAddress, Wri
teAddr, pBuffer, buffersize) != HAL_OK)
00441   {
00442     return EEPROM_FAIL;
00443   }
00444
00445   /* Wait for EEPROM Standby state */
00446   if (EEPROM_I2C_WaitEepromStandbyState() !=
EEPROM_OK)
00447   {
00448     return EEPROM_FAIL;
00449   }
00450
00451   return EEPROM_OK;
00452 }
00453
00454 /**
00455   * @brief  Wait for EEPROM I2C Standby stat
e.
00456   *
00457   * @note  This function allows to wait and
check that EEPROM has finished the
00458   *        last operation. It is mostly used
 after Write operation: after receiving
00459   *        the buffer to be written, the EEP
ROM may need additional time to actually
00460   *        perform the write operation. Duri
ng this time, it doesn't answer to
00461   *        I2C packets addressed to it. Once
 the write operation is complete
00462   *        the EEPROM responds to its addres
s.
00463   *
00464   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00465   *         different from EEPROM_OK (0) or
```

```
the timeout user callback.
00466   */
00467 static uint32_t EEPROM_I2C_WaitEepromStandby
State(void)
00468 {
00469   /* Check if the maximum allowed number of
trials has bee reached */
00470   if (EEPROM_IO_IsDeviceReady(EEPROMAddress,
EEPROM_MAX_TRIALS) != HAL_OK)
00471   {
00472     /* If the maximum number of trials has b
een reached, exit the function */
00473     BSP_EEPROM_TIMEOUT_UserCallback();
00474     return EEPROM_TIMEOUT;
00475   }
00476   return EEPROM_OK;
00477 }
00478
00479 /**
00480   * @}
00481   */
00482
00483 /**
00484   * @}
00485   */
00486
00487 /**
00488   * @}
00489   */
00490
00491 /**
00492   * @}
00493   */
00494
00495 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

Generated on Wed Jul 5 2017 08:56:10 for STM32072B_EVAL BSP
User Manual by  1.7.6.1

STM32072B_EVAL BSP User Manual

Exported Types

STM32072B_EVAL LCD

Data Structures

struct	**LCD_DrawPropTypeDef**

STM32072B_EVAL BSP User Manual

stm32072b_eval_lcd.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32072b_eval_lcd.h
00004   * @author  MCD Application Team
00005   * @brief   This file contains all the functions prototypes for the
00006   *          stm32072b_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 __STM32072B_EVAL_LCD_H
00039 #define __STM32072B_EVAL_LCD_H
00040
00041 #ifdef __cplusplus
00042  extern "C" {
00043 #endif
00044
00045 /* Includes ------------------------------
-------------------------------------*/
00046 #include "stm32072b_eval.h"
00047 /* Include LCD component Driver */
00048 #include "../Components/hx8347d/hx8347d.h"

00049 #include "../Components/spfd5408/spfd5408.h"

00050 #include "../../../Utilities/Fonts/fonts.h"

00051
00052 /** @addtogroup BSP
00053   * @{
00054   */
00055
00056 /** @addtogroup STM32072B_EVAL
00057   * @{
00058   */
00059
00060 /** @defgroup STM32072B_EVAL_LCD STM32072B_E
VAL LCD
00061   * @{
00062   */
00063
00064 /** @defgroup STM32072B_EVAL_LCD_Exported_Ty
pes Exported Types
```

```
00065   * @{
00066   */
00067 typedef struct
00068 {
00069   uint32_t TextColor;
00070   uint32_t BackColor;
00071   sFONT    *pFont;
00072
00073 }LCD_DrawPropTypeDef;
00074 /**
00075   * @}
00076   */
00077
00078 /** @defgroup STM32072B_EVAL_LCD_Exported_Co
nstants Exported Constants
00079   * @{
00080   */
00081 /**
00082   * @brief  LCD status structure definition

00083   */
00084 #define   LCD_OK          0x00
00085 #define   LCD_ERROR       0x01
00086 #define   LCD_TIMEOUT     0x02
00087
00088 typedef struct
00089 {
00090   int16_t X;
00091   int16_t Y;
00092
00093 }Point, * pPoint;
00094
00095 /**
00096   * @brief  Line mode structures definition

00097   */
00098 typedef enum
```

```
00099 {
00100   CENTER_MODE             = 0x01,     /*!< Ce
nter mode */
00101   RIGHT_MODE              = 0x02,     /*!< Ri
ght mode  */
00102   LEFT_MODE               = 0x03      /*!< Le
ft mode   */

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

```
00132 #define LCD_COLOR_BROWN          0xA145
00133 #define LCD_COLOR_ORANGE         0xFD20
00134
00135 /**
00136   * @brief LCD default font
00137   */
00138 #define LCD_DEFAULT_FONT          Font24
00139
00140 /**
00141   * @}
00142   */
00143
00144 /** @defgroup STM32072B_EVAL_LCD_Exported_Fu
nctions Exported Functions
00145   * @{
00146   */
00147 uint8_t  BSP_LCD_Init(void);
00148 uint32_t BSP_LCD_GetXSize(void);
00149 uint32_t BSP_LCD_GetYSize(void);
00150
00151 uint16_t BSP_LCD_GetTextColor(void);
00152 uint16_t BSP_LCD_GetBackColor(void);
00153 void     BSP_LCD_SetTextColor(__IO uint16_t
Color);
00154 void     BSP_LCD_SetBackColor(__IO uint16_t
Color);
00155 void     BSP_LCD_SetFont(sFONT *pFonts);
00156 sFONT    *BSP_LCD_GetFont(void);
00157
00158 void     BSP_LCD_Clear(uint16_t Color);
00159 void     BSP_LCD_ClearStringLine(uint16_t Li
ne);
00160 void     BSP_LCD_DisplayStringAtLine(uint16_
t Line, uint8_t *pText);
00161 void     BSP_LCD_DisplayStringAt(uint16_t Xp
os, uint16_t Ypos, uint8_t *pText, Line_ModeTypdef
 Mode);
```

```
00162 void     BSP_LCD_DisplayChar(uint16_t Xpos,
uint16_t Ypos, uint8_t Ascii);
00163
00164 uint16_t BSP_LCD_ReadPixel(uint16_t Xpos, ui
nt16_t Ypos);
00165 void     BSP_LCD_DrawHLine(uint16_t Xpos, ui
nt16_t Ypos, uint16_t Length);
00166 void     BSP_LCD_DrawVLine(uint16_t Xpos, ui
nt16_t Ypos, uint16_t Length);
00167 void     BSP_LCD_DrawLine(uint16_t X1, uint1
6_t Y1, uint16_t X2, uint16_t Y2);
00168 void     BSP_LCD_DrawRect(uint16_t Xpos, uin
t16_t Ypos, uint16_t Width, uint16_t Height);
00169 void     BSP_LCD_DrawCircle(uint16_t Xpos, u
int16_t Ypos, uint16_t Radius);
00170 void     BSP_LCD_DrawPolygon(pPoint pPoints,
 uint16_t PointCount);
00171 void     BSP_LCD_DrawEllipse(int Xpos, int Y
pos, int XRadius, int YRadius);
00172 void     BSP_LCD_DrawBitmap(uint16_t Xpos, u
int16_t Ypos, uint8_t *pBmp);
00173
00174 void     BSP_LCD_FillRect(uint16_t Xpos, uin
t16_t Ypos, uint16_t Width, uint16_t Height);
00175 void     BSP_LCD_FillCircle(uint16_t Xpos, u
int16_t Ypos, uint16_t Radius);
00176 void     BSP_LCD_FillEllipse(int Xpos, int Y
pos, int XRadius, int YRadius);
00177
00178 void     BSP_LCD_DisplayOff(void);
00179 void     BSP_LCD_DisplayOn(void);
00180
00181 /**
00182   * @}
00183   */
00184
00185 /**
```

```
00186   * @}
00187   */
00188
00189 /**
00190   * @}
00191   */
00192
00193 /**
00194   * @}
00195   */
00196
00197 #ifdef __cplusplus
00198 }
00199 #endif
00200
00201 #endif /* __STM32072B_EVAL_LCD_H */
00202
00203 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM32072B_EVAL BSP User Manual

stm32072b_eval_lcd.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32072b_eval_lcd.c
00004   * @author  MCD Application Team
00005   * @brief   This file includes the driver for Liquid Crystal Display modules
00006   *          mounted on STM32072B-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-240320LDTNQ
W00H (SPFD5408D) and
00043      AM240320LGTNQW00H (HX8347D) LCD mounted
 on MB895 daughter board
00044    - The SPFD5408D and HX8347D components dr
iver MUST be included with this driver.
00045
00046 2. Driver description:
00047 ---------------------
00048   + Initialization steps:
00049      o Initialize the LCD using the LCD_Init
() function.
00050
00051   + Display on LCD
00052      o Clear the hole LCD using yhe LCD_Clea
r() function or only one specified
00053        string line using the LCD_ClearString
Line() function.
00054      o Display a character on the specified
line and column using the LCD_DisplayChar()
00055        function or a complete string line us
ing the LCD_DisplayStringAtLine() function.
00056      o Display a string line on the specifie
d position (x,y in pixel) and align mode
00057        using the LCD_DisplayStringAtLine() f
unction.
```

```
00058      o Draw and fill a basic shapes (dot, li
ne, rectangle, circle, ellipse, .. bitmap)
00059        on LCD using a set of functions.
00060
00061 ------------------------------------------
-----------------------------------*/
00062
00063 /* Includes ---------------------------------
-----------------------------------*/
00064 #include "stm32072b_eval_lcd.h"
00065 #include "../../../Utilities/Fonts/fonts.h"
00066 #include "../../../Utilities/Fonts/font24.c"
00067 #include "../../../Utilities/Fonts/font20.c"
00068 #include "../../../Utilities/Fonts/font16.c"
00069 #include "../../../Utilities/Fonts/font12.c"
00070 #include "../../../Utilities/Fonts/font8.c"
00071
00072 /** @addtogroup BSP
00073   * @{
00074   */
00075
00076 /** @addtogroup STM32072B_EVAL
00077   * @{
00078   */
00079
00080 /** @addtogroup STM32072B_EVAL_LCD
00081   * @{
00082   */
00083
00084 /** @defgroup STM32072B_EVAL_LCD_Private_Con
stants Private Constants
00085   * @{
00086   */
00087 #define POLY_X(Z)              ((int32_t)((
pPoints + (Z))->X))
00088 #define POLY_Y(Z)              ((int32_t)((
pPoints + (Z))->Y))
```

```
00089
00090 #define MAX_HEIGHT_FONT         17
00091 #define MAX_WIDTH_FONT          24
00092 #define OFFSET_BITMAP           54
00093 /**
00094   * @}
00095   */
00096
00097 /** @defgroup STM32072B_EVAL_LCD_Private_Mac
ros Private Macros
00098   * @{
00099   */
00100 #define ABS(X)  ((X) > 0 ? (X) : -(X))
00101
00102 /**
00103   * @}
00104   */
00105
00106 /** @defgroup STM32072B_EVAL_LCD_Private_Var
iables Private Variables
00107   * @{
00108   */
00109 LCD_DrawPropTypeDef DrawProp;
00110
00111 static LCD_DrvTypeDef  *lcd_drv;
00112
00113 /* Max size of bitmap will based on a font24
 (17x24) */
00114 static uint8_t bitmap[MAX_HEIGHT_FONT*MAX_WI
DTH_FONT*2+OFFSET_BITMAP] = {0};
00115
00116 /**
00117   * @}
00118   */
00119
00120 /** @defgroup STM32072B_EVAL_LCD_Private_Fun
ctions Private Functions
```

```
00121   * @{
00122   */
00123 static void LCD_DrawPixel(uint16_t Xpos, uin
t16_t Ypos, uint16_t RGBCode);
00124 static void LCD_DrawChar(uint16_t Xpos, uint
16_t Ypos, const uint8_t *pChar);
00125 static void LCD_SetDisplayWindow(uint16_t Xp
os, uint16_t Ypos, uint16_t Width, uint16_t Height
);
00126 /**
00127   * @}
00128   */
00129
00130 /** @addtogroup STM32072B_EVAL_LCD_Exported_
Functions
00131   * @{
00132   */
00133
00134 /**
00135   * @brief  Initializes the LCD.
00136   * @retval LCD state
00137   */
00138 uint8_t BSP_LCD_Init(void)
00139 {
00140   uint8_t ret = LCD_ERROR;
00141
00142   /* Default value for draw propriety */
00143   DrawProp.BackColor = 0xFFFF;
00144   DrawProp.pFont     = &Font24;
00145   DrawProp.TextColor = 0x0000;
00146
00147   if(spfd5408_drv.ReadID() == SPFD5408_ID)
00148   {
00149     lcd_drv = &spfd5408_drv;
00150     ret = LCD_OK;
00151   }
00152   else
```

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

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

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

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

00279   DrawProp.TextColor = DrawProp.BackColor;;
00280
00281   /* Draw a rectangle with background color
*/
00282   BSP_LCD_FillRect(0, (Line * DrawProp.pFont
->Height), BSP_LCD_GetXSize(), DrawProp.pFont->Hei
ght);
00283
00284   DrawProp.TextColor = colorbackup;
00285   BSP_LCD_SetTextColor(DrawProp.TextColor);
00286 }
00287
00288 /**
```

```
00289   * @brief  Displays one character.
00290   * @param  Xpos Start column address
00291   * @param  Ypos Line where to display the c
haracter shape.
00292   * @param  Ascii Character ascii code
00293   *           This parameter must be a numbe
r between Min_Data = 0x20 and Max_Data = 0x7E
00294   * @retval None
00295   */
00296 void BSP_LCD_DisplayChar(uint16_t Xpos, uint
16_t Ypos, uint8_t Ascii)
00297 {
00298   LCD_DrawChar(Ypos, Xpos, &DrawProp.pFont->
table[(Ascii-' ') *\
00299     DrawProp.pFont->Height * ((DrawProp.pFont
->Width + 7) / 8)]);
00300 }
00301
00302 /**
00303   * @brief  Displays characters on the LCD.
00304   * @param  Xpos X position (in pixel)
00305   * @param  Ypos Y position (in pixel)
00306   * @param  pText Pointer to string to displ
ay on LCD
00307   * @param  Mode Display mode
00308   *           This parameter can be one of th
e following values:
00309   *             @arg  CENTER_MODE
00310   *             @arg  RIGHT_MODE
00311   *             @arg  LEFT_MODE
00312   * @retval None
00313   */
00314 void BSP_LCD_DisplayStringAt(uint16_t Xpos,
uint16_t Ypos, uint8_t *pText, Line_ModeTypdef Mod
e)
00315 {
00316   uint16_t refcolumn = 1, counter = 0;
```

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

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

00356     refcolumn += DrawProp.pFont->Width;
00357     /* Point on the next character */
00358     pText++;
00359     counter++;
00360   }
00361 }
00362
00363 /**
00364   * @brief  Displays a character on the LCD.
00365   * @param  Line Line where to display the c
haracter shape
00366   *          This parameter can be one of th
e following values:
00367   *            @arg  0..9: if the Current fo
nts is Font16x24
00368   *            @arg  0..19: if the Current f
onts is Font12x12 or Font8x12
00369   *            @arg  0..29: if the Current f
onts is Font8x8
00370   * @param  pText Pointer to string to displ
ay on LCD
00371   * @retval None
00372   */
00373 void BSP_LCD_DisplayStringAtLine(uint16_t Li
ne, uint8_t *pText)
00374 {
00375   BSP_LCD_DisplayStringAt(0, LINE(Line),pTex
```

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

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

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

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

```
00497   {
00498     xinc2 = 0;                  /* Don't cha
nge the x for every iteration */
00499     yinc1 = 0;                  /* Don't cha
nge the y when numerator >= denominator */
00500     den = deltay;
00501     num = deltay / 2;
00502     numadd = deltax;
00503     numpixels = deltay;         /* There are
 more y-values than x-values */
00504   }
00505
00506   for (curpixel = 0; curpixel <= numpixels;
curpixel++)
00507   {
00508     LCD_DrawPixel(x, y, DrawProp.TextColor);
  /* Draw the current pixel */
00509     num += numadd;
  /* Increase the numerator by the top of the frac
tion */
00510     if (num >= den)
  /* Check if numerator >= denominator */
00511     {
00512       num -= den;
  /* Calculate the new numerator value */
00513       x += xinc1;
  /* Change the x as appropriate */
00514       y += yinc1;
  /* Change the y as appropriate */
00515     }
00516     x += xinc2;
  /* Change the x as appropriate */
00517     y += yinc2;
  /* Change the y as appropriate */
00518   }
00519 }
00520
```

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

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

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

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

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

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

00701   }
00702   while(Height--);
00703 }
00704
00705 /**
00706   * @brief  Draws a full circle.
```

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

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

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

```
00808   */
00809
00810 /** @defgroup STM32072B_EVAL_LCD_Private_Fun
ctions Private Functions
00811   * @{
00812   */
00813 /*****************************************
**********************************
00814                                   Static Function
00815 *****************************************
**********************************/
00816 /**
00817   * @brief  Draws a pixel on LCD.
00818   * @param  Xpos X position
00819   * @param  Ypos Y position
00820   * @param  RGBCode Pixel color in RGB mode
(5-6-5)
00821   * @retval None
00822   */
00823 static void LCD_DrawPixel(uint16_t Xpos, uin
t16_t Ypos, uint16_t RGBCode)
00824 {
00825   if(lcd_drv->WritePixel != NULL)
00826   {
00827     lcd_drv->WritePixel(Xpos, Ypos, RGBCode)
;
00828   }
00829 }
00830
00831 /**
00832   * @brief  Draws a character on LCD.
00833   * @param  Xpos Line where to display the c
haracter shape
00834   * @param  Ypos Start column address
00835   * @param  pChar Pointer to the character d
ata
00836   * @retval None
```

```
00837   */
00838 static void LCD_DrawChar(uint16_t Xpos, uint
16_t Ypos, const uint8_t *pChar)
00839 {
00840   uint32_t counterh = 0, counterw = 0, index
 = 0;
00841   uint16_t height = 0, width = 0;
00842   uint8_t offset = 0;
00843   uint8_t *pchar = NULL;
00844   uint32_t line = 0;
00845
00846   height = DrawProp.pFont->Height;
00847   width  = DrawProp.pFont->Width;
00848
00849   /* Fill bitmap header*/
00850   *(uint16_t *) (bitmap + 2) = (uint16_t)(he
ight*width*2+OFFSET_BITMAP);
00851   *(uint16_t *) (bitmap + 4) = (uint16_t)((h
eight*width*2+OFFSET_BITMAP)>>16);
00852   *(uint16_t *) (bitmap + 10) = OFFSET_BITMAP
;
00853   *(uint16_t *) (bitmap + 18) = (uint16_t)(w
idth);
00854   *(uint16_t *) (bitmap + 20) = (uint16_t)((
width)>>16);
00855   *(uint16_t *) (bitmap + 22) = (uint16_t)(h
eight);
00856   *(uint16_t *) (bitmap + 24) = (uint16_t)((
height)>>16);
00857
00858   offset =  8 *((width + 7)/8) -  width ;
00859
00860   for(counterh = 0; counterh < height; count
erh++)
00861   {
00862     pchar = ((uint8_t *)pChar + (width + 7)/
8 * counterh);
```

```
00863
00864     if(((width + 7)/8) == 3)
00865     {
00866       line =  (pchar[0]<< 16) | (pchar[1]<<
8) | pchar[2];
00867     }
00868
00869     if(((width + 7)/8) == 2)
00870     {
00871       line =  (pchar[0]<< 8) | pchar[1];
00872     }
00873
00874     if(((width + 7)/8) == 1)
00875     {
00876       line =  pchar[0];
00877     }
00878
00879     for (counterw = 0; counterw < width; cou
nterw++)
00880     {
00881       /* Image in the bitmap is written from
 the bottom to the top */
00882       /* Need to invert image in the bitmap
*/
00883       index = (((height-counterh-1)*width)+(
counterw))*2+OFFSET_BITMAP;
00884       if(line & (1 << (width- counterw + off
set- 1)))
00885       {
00886         bitmap[index] = (uint8_t)DrawProp.Te
xtColor;
00887         bitmap[index+1] = (uint8_t)(DrawProp.
TextColor >> 8);
00888       }
00889       else
00890       {
00891         bitmap[index] = (uint8_t)DrawProp.Ba
```

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

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

STM32072B_EVAL BSP User Manual

Exported Constants

STM32072B_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 131 of file stm32072b_eval_lcd.h.

#define LCD_COLOR_BLUE 0x001F

LCD color.

Definition at line 109 of file stm32072b_eval_lcd.h.

#define LCD_COLOR_BROWN 0xA145

Definition at line 132 of file stm32072b_eval_lcd.h.

#define LCD_COLOR_CYAN 0x07FF

Definition at line 112 of file stm32072b_eval_lcd.h.

#define LCD_COLOR_DARKBLUE 0x0010

Definition at line 121 of file stm32072b_eval_lcd.h.

#define LCD_COLOR_DARKCYAN 0x0410

Definition at line 124 of file stm32072b_eval_lcd.h.

#define LCD_COLOR_DARKGRAY 0x4208

Definition at line **130** of file **stm32072b_eval_lcd.h**.

#define LCD_COLOR_DARKGREEN 0x0400

Definition at line **122** of file **stm32072b_eval_lcd.h**.

#define LCD_COLOR_DARKMAGENTA 0x8010

Definition at line **125** of file **stm32072b_eval_lcd.h**.

#define LCD_COLOR_DARKRED 0x8000

Definition at line **123** of file **stm32072b_eval_lcd.h**.

#define LCD_COLOR_DARKYELLOW 0x8400

Definition at line **126** of file **stm32072b_eval_lcd.h**.

#define LCD_COLOR_GRAY 0x8410

Definition at line **129** of file **stm32072b_eval_lcd.h**.

#define LCD_COLOR_GREEN 0x07E0

Definition at line **110** of file **stm32072b_eval_lcd.h**.

#define LCD_COLOR_LIGHTBLUE 0x841F

Definition at line **115** of file **stm32072b_eval_lcd.h**.

#define LCD_COLOR_LIGHTCYAN 0x87FF

Definition at line 118 of file stm32072b_eval_lcd.h.

#define LCD_COLOR_LIGHTGRAY 0xD69A

Definition at line 128 of file stm32072b_eval_lcd.h.

#define LCD_COLOR_LIGHTGREEN 0x87F0

Definition at line 116 of file stm32072b_eval_lcd.h.

#define LCD_COLOR_LIGHTMAGENTA 0xFC1F

Definition at line 119 of file stm32072b_eval_lcd.h.

#define LCD_COLOR_LIGHTRED 0xFC10

Definition at line 117 of file stm32072b_eval_lcd.h.

#define LCD_COLOR_LIGHTYELLOW 0xFFF0

Definition at line 120 of file stm32072b_eval_lcd.h.

#define LCD_COLOR_MAGENTA 0xF81F

Definition at line 113 of file stm32072b_eval_lcd.h.

#define LCD_COLOR_ORANGE 0xFD20

Definition at line 133 of file stm32072b_eval_lcd.h.

#define LCD_COLOR_RED 0xF800

Definition at line 111 of file stm32072b_eval_lcd.h.

#define LCD_COLOR_WHITE 0xFFFF

Definition at line 127 of file stm32072b_eval_lcd.h.

#define LCD_COLOR_YELLOW 0xFFE0

Definition at line 114 of file stm32072b_eval_lcd.h.

#define LCD_DEFAULT_FONT Font24

LCD default font.

Definition at line 138 of file stm32072b_eval_lcd.h.

Referenced by BSP_LCD_DrawCircle(), and BSP_LCD_Init().

#define LCD_ERROR 0x01

Definition at line 85 of file stm32072b_eval_lcd.h.

Referenced by BSP_LCD_Init().

#define LCD_OK 0x00

LCD status structure definition.

Definition at line **84** of file **stm32072b_eval_lcd.h**.

Referenced by **BSP_LCD_Init()**.

#define LCD_TIMEOUT 0x02

Definition at line **86** of file **stm32072b_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 **98** of file **stm32072b_eval_lcd.h**.

STM32072B_EVAL BSP User Manual

Types Definitions

STM32072B_EVAL SD

Data Structures

struct	**SD_CmdAnswer_typedef**

STM32072B_EVAL BSP User Manual

stm32072b_eval_sd.c

Go to the documentation of this file.

```
00001 /**
00002 ******************************************************************************
00003 * @file    stm32072b_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 STM32072B-EVAL board.
00007 *          It implements a high level communication layer for read and write
00008 *          from/to this memory. The needed STM32F0xx hardware resources (SPI and
00009 *          GPIO) are defined in stm32072b_eval.h file, and the initialization is
00010 *          performed in SD_IO_Init() function declared in stm32072b_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
```

```
00014 *          SD_IO_Init() function.
00015 *
00016 *          +-------------------------------
----------------------+
00017 *          |                         Pin assignm
ent                    |
00018 *          +-------------------------+------
---------+-------------+
00019 *          |  STM32F0xx SPI Pins     |     S
D        |    Pin      |
00020 *          +-------------------------+------
---------+-------------+
00021 *          | SD_SPI_CS_PIN           |   Chi
pSelect  |    1        |
00022 *          | SD_SPI_MOSI_PIN / MOSI  |   Dat
aIn      |    2        |
00023 *          |                         |   GND
         |    3 (0 V)  |
00024 *          |                         |   VDD
         |    4 (3.3 V)|
00025 *          | SD_SPI_SCK_PIN / SCLK   |   Clo
ck       |    5        |
00026 *          |                         |   GND
         |    6 (0 V)  |
00027 *          | SD_SPI_MISO_PIN / MISO  |   Dat
aOut     |    7        |
00028 *          +-------------------------+------
---------+-------------+
00029 ******************************************
**********************************
00030 * @attention
00031 *
00032 * <h2><center>&copy; COPYRIGHT(c) 2016 STMic
roelectronics</center></h2>
00033 *
00034 * Redistribution and use in source and binar
y forms, with or without modification,
```

```
00035 * are permitted provided that the following
conditions are met:
00036 *   1. Redistributions of source code must r
etain the above copyright notice,
00037 *      this list of conditions and the follo
wing disclaimer.
00038 *   2. Redistributions in binary form must r
eproduce the above copyright notice,
00039 *      this list of conditions and the follo
wing disclaimer in the documentation
00040 *      and/or other materials provided with
the distribution.
00041 *   3. Neither the name of STMicroelectronic
s nor the names of its contributors
00042 *      may be used to endorse or promote pro
ducts derived from this software
00043 *      without specific prior written permis
sion.
00044 *
00045 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
 HOLDERS AND CONTRIBUTORS "AS IS"
00046 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INC
LUDING, BUT NOT LIMITED TO, THE
00047 * IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS FOR A PARTICULAR PURPOSE ARE
00048 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGH
T HOLDER OR CONTRIBUTORS BE LIABLE
00049 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPEC
IAL, EXEMPLARY, OR CONSEQUENTIAL
00050 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PR
OCUREMENT OF SUBSTITUTE GOODS OR
00051 * SERVICES; LOSS OF USE, DATA, OR PROFITS; O
R BUSINESS INTERRUPTION) HOWEVER
00052 * CAUSED AND ON ANY THEORY OF LIABILITY, WHE
THER IN CONTRACT, STRICT LIABILITY,
00053 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE
) ARISING IN ANY WAY OUT OF THE USE
```

```
00054 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE P
OSSIBILITY OF SUCH DAMAGE.
00055 *
00056 ****************************************
*********************************
00057 @verbatim
00058   ========================================
=================================
00059                        ##### How to use this d
river #####
00060   ========================================
=================================
00061   [..]
00062    (#) This driver is used to drive the micr
o SD external card mounted on STM32072B-EVAL
00063        evaluation board.
00064    (#) This driver does not need a specific
component driver for the micro SD device
00065        to be included with.
00066
00067    (#) Initialization steps:
00068        (++) Initialize the micro SD card usi
ng the SD_Init() function.
00069        (++) To check the SD card presence yo
u can use the function BSP_SD_IsDetected() which
00070             returns the detection status
00071        (++) The function BSP_SD_GetCardInfo(
) is used to get the micro SD card information
00072             which is stored in the structure
 "SD_CardInfo".
00073
00074    (#) Micro SD card operations
00075        (++) The micro SD card can be accesse
d with read/write block(s) operations once
00076             it is reay for access. The acces
s cand be performed in polling
00077             mode by calling the functions SD
```

```
_ReadBlocks()/SD_WriteBlocks()
00078         (++) The SD erase block(s) is perform
ed using the function BSP_SD_Erase() with
00079              specifying the number of blocks
to erase.
00080         (++) The SD runtime status is returne
d when calling the function BSP_SD_GetStatus().
00081
00082   @endverbatim
00083 */
00084
00085
00086 /* Includes ------------------------------
-----------------------------------*/
00087 #include "stm32072b_eval_sd.h"
00088 #include "stm32f0xx_hal.h"
00089 #include "stdlib.h"
00090 #include "string.h"
00091 #include "stdio.h"
00092
00093 /** @addtogroup BSP
00094 * @{
00095 */
00096
00097 /** @addtogroup STM32072B_EVAL
00098 * @{
00099 */
00100
00101 /** @addtogroup STM32072B_EVAL_SD
00102 * @{
00103 */
00104
00105 /* Private typedef -----------------------
-----------------------------------*/
00106
00107 /** @defgroup STM32072BEVAL_SD_Private_Types
_Definitions Types Definitions
```

```
00108   * @{
00109   */
00110 typedef struct {
00111   uint8_t r1;
00112   uint8_t r2;
00113   uint8_t r3;
00114   uint8_t r4;
00115   uint8_t r5;
00116 } SD_CmdAnswer_typedef;
00117
00118 /**
00119   * @}
00120   */
00121
00122 /* Private define ------------------------
-----------------------------------*/
00123
00124 /** @defgroup STM32072B_EVAL_SD_Private_Cons
tants Private Constants
00125 * @{
00126 */
00127 #define SD_DUMMY_BYTE            0xFF
00128
00129 #define SD_MAX_FRAME_LENGTH        17    /*
Lenght = 16 + 1 */
00130 #define SD_CMD_LENGTH               6
00131
00132 #define SD_MAX_TRY                100    /*
Number of try */
00133
00134 #define SD_CSD_STRUCT_V1          0x2    /*
CSD struct version V1 */
00135 #define SD_CSD_STRUCT_V2          0x1    /*
CSD struct version V2 */
00136
00137
00138 /**
```

```
00139   * @brief  SD ansewer format
00140   */
00141 typedef enum {
00142  SD_ANSWER_R1_EXPECTED,
00143  SD_ANSWER_R1B_EXPECTED,
00144  SD_ANSWER_R2_EXPECTED,
00145  SD_ANSWER_R3_EXPECTED,
00146  SD_ANSWER_R4R5_EXPECTED,
00147  SD_ANSWER_R7_EXPECTED,
00148 }SD_Answer_type;
00149
00150 /**
00151   * @brief  Start Data tokens:
00152   *         Tokens (necessary because at nop
/idle (and CS active) only 0xff is
00153   *         on the data/command line)
00154   */
00155 #define SD_TOKEN_START_DATA_SINGLE_BLOCK_REA
D    0xFE  /* Data token start byte, Start Single
Block Read */
00156 #define SD_TOKEN_START_DATA_MULTIPLE_BLOCK_R
EAD  0xFE  /* Data token start byte, Start Multipl
e Block Read */
00157 #define SD_TOKEN_START_DATA_SINGLE_BLOCK_WRI
TE   0xFE  /* Data token start byte, Start Single
Block Write */
00158 #define SD_TOKEN_START_DATA_MULTIPLE_BLOCK_W
RITE 0xFD  /* Data token start byte, Start Multipl
e Block Write */
00159 #define SD_TOKEN_STOP_DATA_MULTIPLE_BLOCK_WR
ITE  0xFD  /* Data toke stop byte, Stop Multiple B
lock Write */
00160
00161 /**
00162   * @brief  Commands: CMDxx = CMD-number | 0
x40
00163   */
```

```
00164 #define SD_CMD_GO_IDLE_STATE          0   /*
 CMD0 = 0x40  */
00165 #define SD_CMD_SEND_OP_COND           1   /*
 CMD1 = 0x41  */
00166 #define SD_CMD_SEND_IF_COND           8   /*
 CMD8 = 0x48  */
00167 #define SD_CMD_SEND_CSD               9   /*
 CMD9 = 0x49  */
00168 #define SD_CMD_SEND_CID               10  /*
 CMD10 = 0x4A */
00169 #define SD_CMD_STOP_TRANSMISSION      12  /*
 CMD12 = 0x4C */
00170 #define SD_CMD_SEND_STATUS            13  /*
 CMD13 = 0x4D */
00171 #define SD_CMD_SET_BLOCKLEN           16  /*
 CMD16 = 0x50 */
00172 #define SD_CMD_READ_SINGLE_BLOCK      17  /*
 CMD17 = 0x51 */
00173 #define SD_CMD_READ_MULT_BLOCK        18  /*
 CMD18 = 0x52 */
00174 #define SD_CMD_SET_BLOCK_COUNT        23  /*
 CMD23 = 0x57 */
00175 #define SD_CMD_WRITE_SINGLE_BLOCK     24  /*
 CMD24 = 0x58 */
00176 #define SD_CMD_WRITE_MULT_BLOCK       25  /*
 CMD25 = 0x59 */
00177 #define SD_CMD_PROG_CSD               27  /*
 CMD27 = 0x5B */
00178 #define SD_CMD_SET_WRITE_PROT         28  /*
 CMD28 = 0x5C */
00179 #define SD_CMD_CLR_WRITE_PROT         29  /*
 CMD29 = 0x5D */
00180 #define SD_CMD_SEND_WRITE_PROT        30  /*
 CMD30 = 0x5E */
00181 #define SD_CMD_SD_ERASE_GRP_START     32  /*
 CMD32 = 0x60 */
00182 #define SD_CMD_SD_ERASE_GRP_END       33  /*
```

```
 CMD33 = 0x61 */
00183 #define SD_CMD_UNTAG_SECTOR             34  /*
 CMD34 = 0x62 */
00184 #define SD_CMD_ERASE_GRP_START          35  /*
 CMD35 = 0x63 */
00185 #define SD_CMD_ERASE_GRP_END            36  /*
 CMD36 = 0x64 */
00186 #define SD_CMD_UNTAG_ERASE_GROUP        37  /*
 CMD37 = 0x65 */
00187 #define SD_CMD_ERASE                    38  /*
 CMD38 = 0x66 */
00188 #define SD_CMD_SD_APP_OP_COND           41  /*
 CMD41 = 0x69 */
00189 #define SD_CMD_APP_CMD                  55  /*
 CMD55 = 0x77 */
00190 #define SD_CMD_READ_OCR                 58  /*
 CMD55 = 0x79 */
00191
00192 /**
00193   * @brief  SD reponses and error flags
00194   */
00195 typedef enum
00196 {
00197 /* R1 answer value */
00198   SD_R1_NO_ERROR              = (0x00),
00199   SD_R1_IN_IDLE_STATE         = (0x01),
00200   SD_R1_ERASE_RESET           = (0x02),
00201   SD_R1_ILLEGAL_COMMAND       = (0x04),
00202   SD_R1_COM_CRC_ERROR         = (0x08),
00203   SD_R1_ERASE_SEQUENCE_ERROR= (0x10),
00204   SD_R1_ADDRESS_ERROR         = (0x20),
00205   SD_R1_PARAMETER_ERROR       = (0x40),
00206
00207 /* R2 answer value */
00208   SD_R2_NO_ERROR              = 0x00,
00209   SD_R2_CARD_LOCKED           = 0x01,
00210   SD_R2_LOCKUNLOCK_ERROR      = 0x02,
```

```
00211   SD_R2_ERROR                  = 0x04,
00212   SD_R2_CC_ERROR               = 0x08,
00213   SD_R2_CARD_ECC_FAILED        = 0x10,
00214   SD_R2_WP_VIOLATION           = 0x20,
00215   SD_R2_ERASE_PARAM            = 0x40,
00216   SD_R2_OUTOFRANGE             = 0x80,
00217
00218 /**
00219   * @brief  Data response error
00220   */
00221   SD_DATA_OK                   = (0x05),
00222   SD_DATA_CRC_ERROR            = (0x0B),
00223   SD_DATA_WRITE_ERROR          = (0x0D),
00224   SD_DATA_OTHER_ERROR          = (0xFF)
00225 } SD_Error;
00226
00227 /**
00228 * @}
00229 */
00230
00231 /* Private macro -------------------------
------------------------------------*/
00232
00233 /* Private variables ---------------------
------------------------------------*/
00234
00235 /** @defgroup STM32072B_EVAL_SD_Private_Vari
ables Private Variables
00236 * @{
00237 */
00238 __IO uint8_t SdStatus = SD_NOT_PRESENT;
00239
00240 /* flag_SDHC :
00241       0 :  Standard capacity
00242       1 : High capacity
00243 */
00244 uint16_t flag_SDHC = 0;
```

```
00245
00246 /**
00247 * @}
00248 */
00249
00250 /* Private function prototypes -----------
-----------------------------------*/
00251 /** @defgroup STM32072B_EVAL_SD_Private_Func
tions Private Functions
00252 * @{
00253 */
00254 static uint8_t SD_GetCIDRegister(SD_CID* Cid
);
00255 static uint8_t SD_GetCSDRegister(SD_CSD* Csd
);
00256 static uint8_t SD_GetDataResponse(void);
00257 static uint8_t SD_GoIdleState(void);
00258 static SD_CmdAnswer_typedef SD_SendCmd(uint8
_t Cmd, uint32_t Arg, uint8_t Crc, uint8_t Answer)
;
00259 static uint8_t SD_WaitData(uint8_t data);
00260 static uint8_t SD_ReadData(void);
00261 /**
00262 * @}
00263 */
00264
00265 /* Exported functions ---------------------
-----------------------------------*/
00266
00267 /** @addtogroup STM32072B_EVAL_SD_Exported_F
unctions
00268 * @{
00269 */
00270
00271 /**
00272   * @brief  Initializes the SD/SD communicat
ion.
```

```
00273   * @retval The SD Response:
00274   *         - MSD_ERROR: Sequence failed
00275   *         - MSD_OK: Sequence succeed
00276   */
00277 uint8_t BSP_SD_Init(void)
00278 {
00279   /* Configure IO functionalities for SD pin
 */
00280   SD_IO_Init();
00281
00282   /* Check SD card detect pin */
00283   if(BSP_SD_IsDetected()==SD_NOT_PRESENT)
00284   {
00285     SdStatus = SD_NOT_PRESENT;
00286     return MSD_ERROR;
00287   }
00288   else
00289   {
00290     SdStatus = SD_PRESENT;
00291   }
00292
00293   /* SD initialized and set to SPI mode prop
erly */
00294   return (SD_GoIdleState());
00295 }
00296
00297 /**
00298  * @brief  Detects if SD card is correctly p
lugged in the memory slot or not.
00299  * @retval Returns if SD is detected or not
00300  */
00301 uint8_t BSP_SD_IsDetected(void)
00302 {
00303   __IO uint8_t status = SD_PRESENT;
00304
00305   /* Check SD card detect pin */
00306   if(HAL_GPIO_ReadPin(SD_DETECT_GPIO_PORT, S
```

```
D_DETECT_PIN) != GPIO_PIN_RESET)
00307   {
00308     status = SD_NOT_PRESENT;
00309   }
00310
00311   return status;
00312 }
00313
00314 /**
00315   * @brief  Returns information about specif
ic card.
00316   * @param  pCardInfo Pointer to a SD_CardIn
fo structure that contains all SD
00317   *         card information.
00318   * @retval The SD Response:
00319   *         - MSD_ERROR: Sequence failed
00320   *         - MSD_OK: Sequence succeed
00321   */
00322 uint8_t BSP_SD_GetCardInfo(SD_CardInfo *pCar
dInfo)
00323 {
00324   uint8_t status;
00325
00326   status = SD_GetCSDRegister(&(pCardInfo->Csd
));
00327   status|= SD_GetCIDRegister(&(pCardInfo->Cid
));
00328   if(flag_SDHC == 1 )
00329   {
00330     pCardInfo->CardBlockSize = 512;
00331     pCardInfo->CardCapacity = (pCardInfo->Csd
.version.v2.DeviceSize + 1) * pCardInfo->CardBlock
Size;
00332   }
00333   else
00334   {
00335     pCardInfo->CardCapacity = (pCardInfo->Csd
```

```
.version.v1.DeviceSize + 1) ;
00336     pCardInfo->CardCapacity *= (1 << (pCardI
nfo->Csd.version.v1.DeviceSizeMul + 2));
00337     pCardInfo->CardBlockSize = 1 << (pCardIn
fo->Csd.RdBlockLen);
00338     pCardInfo->CardCapacity *= pCardInfo->Ca
rdBlockSize;
00339   }
00340
00341   return status;
00342 }
00343
00344 /**
00345   * @brief  Reads block(s) from a specified
address in the SD card, in polling mode.
00346   * @param  pData Pointer to the buffer that
 will contain the data to transmit
00347   * @param  ReadAddr Address from where data
 is to be read
00348   * @param  BlockSize SD card data block siz
e, that should be 512
00349   * @param  NumberOfBlocks Number of SD bloc
ks to read
00350   * @retval SD status
00351   */
00352 uint8_t BSP_SD_ReadBlocks(uint32_t* pData, u
int32_t ReadAddr, uint16_t BlockSize, uint32_t Num
berOfBlocks)
00353 {
00354   uint32_t offset = 0;
00355   uint8_t retr = BSP_SD_ERROR;
00356   uint8_t *ptr = NULL;
00357   SD_CmdAnswer_typedef response;
00358
00359   /* Send CMD16 (SD_CMD_SET_BLOCKLEN) to set
 the size of the block and
00360     Check if the SD acknowledged the set bl
```

```
ock length command: R1 response (0x00: no errors)
*/
00361   response = SD_SendCmd(SD_CMD_SET_BLOCKLEN,
 BlockSize, 0xFF, SD_ANSWER_R1_EXPECTED);
00362   SD_IO_CSState(1);
00363   SD_IO_WriteByte(SD_DUMMY_BYTE);
00364   if ( response.r1 != SD_R1_NO_ERROR)
00365   {
00366      goto error;
00367   }
00368
00369   ptr = malloc(sizeof(uint8_t)*BlockSize);
00370   if( ptr == NULL )
00371   {
00372      goto error;
00373   }
00374   memset(ptr, SD_DUMMY_BYTE, sizeof(uint8_t)
*BlockSize);
00375
00376   /* Data transfer */
00377   while (NumberOfBlocks--)
00378   {
00379     /* Send CMD17 (SD_CMD_READ_SINGLE_BLOCK)
 to read one block */
00380     /* Check if the SD acknowledged the read
 block command: R1 response (0x00: no errors) */
00381     response = SD_SendCmd(SD_CMD_READ_SINGLE
_BLOCK, (ReadAddr + offset)/(flag_SDHC == 1 ?Block
Size: 1), 0xFF, SD_ANSWER_R1_EXPECTED);
00382     if ( response.r1 != SD_R1_NO_ERROR)
00383     {
00384       goto error;
00385     }
00386
00387     /* Now look for the data token to signif
y the start of the data */
00388     if (SD_WaitData(SD_TOKEN_START_DATA_SING
```

```
LE_BLOCK_READ) == BSP_SD_OK)
00389     {
00390       /* Read the SD block data : read NumBy
teToRead data */
00391       SD_IO_WriteReadData(ptr, (uint8_t*)pDa
ta + offset, BlockSize);
00392
00393       /* Set next read address*/
00394       offset += BlockSize;
00395       /* get CRC bytes (not really needed by
 us, but required by SD) */
00396       SD_IO_WriteByte(SD_DUMMY_BYTE);
00397       SD_IO_WriteByte(SD_DUMMY_BYTE);
00398     }
00399     else
00400     {
00401       goto error;
00402     }
00403
00404     /* End the command data read cycle */
00405     SD_IO_CSState(1);
00406     SD_IO_WriteByte(SD_DUMMY_BYTE);
00407   }
00408
00409   retr = BSP_SD_OK;
00410
00411 error :
00412   /* Send dummy byte: 8 Clock pulses of dela
y */
00413   SD_IO_CSState(1);
00414   SD_IO_WriteByte(SD_DUMMY_BYTE);
00415   if(ptr != NULL) free(ptr);
00416
00417   /* Return the reponse */
00418   return retr;
00419 }
00420
```

```
00421 /**
00422   * @brief  Writes block(s) to a specified a
ddress in the SD card, in polling mode.
00423   * @param  pData Pointer to the buffer that
 will contain the data to transmit
00424   * @param  WriteAddr Address from where dat
a is to be written
00425   * @param  BlockSize SD card data block siz
e, that should be 512
00426   * @param  NumberOfBlocks Number of SD bloc
ks to write
00427   * @retval SD status
00428   */
00429 uint8_t BSP_SD_WriteBlocks(uint32_t* pData,
uint32_t WriteAddr, uint16_t BlockSize, uint32_t N
umberOfBlocks)
00430 {
00431   uint32_t offset = 0;
00432   uint8_t retr = BSP_SD_ERROR;
00433   uint8_t *ptr = NULL;
00434   SD_CmdAnswer_typedef response;
00435
00436   /* Send CMD16 (SD_CMD_SET_BLOCKLEN) to set
 the size of the block and
00437      Check if the SD acknowledged the set bl
ock length command: R1 response (0x00: no errors)
*/
00438   response = SD_SendCmd(SD_CMD_SET_BLOCKLEN,
 BlockSize, 0xFF, SD_ANSWER_R1_EXPECTED);
00439   SD_IO_CSState(1);
00440   SD_IO_WriteByte(SD_DUMMY_BYTE);
00441   if ( response.r1 != SD_R1_NO_ERROR)
00442   {
00443     goto error;
00444   }
00445
00446   ptr = malloc(sizeof(uint8_t)*BlockSize);
```

```
00447   if (ptr == NULL)
00448   {
00449     goto error;
00450   }
00451
00452   /* Data transfer */
00453   while (NumberOfBlocks--)
00454   {
00455     /* Send CMD24 (SD_CMD_WRITE_SINGLE_BLOCK
) to write blocks  and
00456        Check if the SD acknowledged the writ
e block command: R1 response (0x00: no errors) */
00457     response = SD_SendCmd(SD_CMD_WRITE_SINGL
E_BLOCK, (WriteAddr + offset)/(flag_SDHC == 1 ? Bl
ockSize: 1), 0xFF, SD_ANSWER_R1_EXPECTED);
00458     if (response.r1 != SD_R1_NO_ERROR)
00459     {
00460       goto error;
00461     }
00462
00463     /* Send dummy byte for NWR timing : one
byte between CMDWRITE and TOKEN */
00464     SD_IO_WriteByte(SD_DUMMY_BYTE);
00465     SD_IO_WriteByte(SD_DUMMY_BYTE);
00466
00467     /* Send the data token to signify the st
art of the data */
00468     SD_IO_WriteByte(SD_TOKEN_START_DATA_SING
LE_BLOCK_WRITE);
00469
00470     /* Write the block data to SD */
00471     SD_IO_WriteReadData((uint8_t*)pData + of
fset, ptr, BlockSize);
00472
00473     /* Set next write address */
00474     offset += BlockSize;
00475
```

```
00476     /* Put CRC bytes (not really needed by u
s, but required by SD) */
00477     SD_IO_WriteByte(SD_DUMMY_BYTE);
00478     SD_IO_WriteByte(SD_DUMMY_BYTE);
00479
00480     /* Read data response */
00481     if (SD_GetDataResponse() != SD_DATA_OK)
00482     {
00483       /* Set response value to failure */
00484       goto error;
00485     }
00486
00487     SD_IO_CSState(1);
00488     SD_IO_WriteByte(SD_DUMMY_BYTE);
00489   }
00490   retr = BSP_SD_OK;
00491
00492 error :
00493   if(ptr != NULL) free(ptr);
00494   /* Send dummy byte: 8 Clock pulses of dela
y */
00495   SD_IO_CSState(1);
00496   SD_IO_WriteByte(SD_DUMMY_BYTE);
00497
00498   /* Return the reponse */
00499   return retr;
00500 }
00501
00502 /**
00503   * @brief  Erases the specified memory area
 of the given SD card.
00504   * @param  StartAddr Start byte address
00505   * @param  EndAddr End byte address
00506   * @retval SD status
00507   */
00508 uint8_t BSP_SD_Erase(uint32_t StartAddr, uin
t32_t EndAddr)
```

```
00509 {
00510   uint8_t retr = BSP_SD_ERROR;
00511   SD_CmdAnswer_typedef response;
00512
00513   /* Send CMD32 (Erase group start) and chec
k if the SD acknowledged the erase command: R1 res
ponse (0x00: no errors) */
00514   response = SD_SendCmd(SD_CMD_SD_ERASE_GRP_
START, StartAddr, 0xFF, SD_ANSWER_R1_EXPECTED);
00515   SD_IO_CSState(1);
00516   SD_IO_WriteByte(SD_DUMMY_BYTE);  if (respo
nse.r1 == SD_R1_NO_ERROR)
00517   {
00518     /* Send CMD33 (Erase group end) and Chec
k if the SD acknowledged the erase command: R1 res
ponse (0x00: no errors) */
00519     response = SD_SendCmd(SD_CMD_SD_ERASE_GR
P_END, EndAddr, 0xFF, SD_ANSWER_R1_EXPECTED);
00520     SD_IO_CSState(1);
00521     SD_IO_WriteByte(SD_DUMMY_BYTE);
00522     if (response.r1 == SD_R1_NO_ERROR)
00523     {
00524       /* Send CMD38 (Erase) and Check if the
 SD acknowledged the erase command: R1 response (0
x00: no errors) */
00525       response = SD_SendCmd(SD_CMD_ERASE, 0,
 0xFF, SD_ANSWER_R1B_EXPECTED);
00526       if (response.r1 == SD_R1_NO_ERROR)
00527       {
00528         retr = BSP_SD_OK;
00529       }
00530       SD_IO_CSState(1);
00531       SD_IO_WriteByte(SD_DUMMY_BYTE);
00532     }
00533   }
00534
00535   /* Return the reponse */
```

```
00536   return retr;
00537 }
00538
00539 /**
00540   * @brief  Returns the SD status.
00541   * @retval The SD status.
00542   */
00543 uint8_t BSP_SD_GetStatus(void)
00544 {
00545   SD_CmdAnswer_typedef retr;
00546
00547   /* Send CMD13 (SD_SEND_STATUS) to get SD s
tatus */
00548   retr = SD_SendCmd(SD_CMD_SEND_STATUS, 0, 0
xFF, SD_ANSWER_R2_EXPECTED);
00549   SD_IO_CSState(1);
00550   SD_IO_WriteByte(SD_DUMMY_BYTE);
00551
00552   /* Find SD status according to card state
*/
00553   if(( retr.r1 == SD_R1_NO_ERROR) && ( retr.
r2 == SD_R2_NO_ERROR))
00554   {
00555     return BSP_SD_OK;
00556   }
00557
00558   return BSP_SD_ERROR;
00559 }
00560
00561 /**
00562   * @brief  Reads the SD card SCD register.
00563   *         Reading the contents of the CSD
register in SPI mode is a simple
00564   *         read-block transaction.
00565   * @param  Csd pointer on an SCD register s
tructure
00566   * @retval SD status
```

```
00567   */
00568 uint8_t SD_GetCSDRegister(SD_CSD* Csd)
00569 {
00570   uint16_t counter = 0;
00571   uint8_t CSD_Tab[16];
00572   uint8_t retr = BSP_SD_ERROR;
00573   SD_CmdAnswer_typedef response;
00574
00575   /* Send CMD9 (CSD register) or CMD10(CSD r
egister) and Wait for response in the R1 format (0
x00 is no errors) */
00576   response = SD_SendCmd(SD_CMD_SEND_CSD, 0,
0xFF, SD_ANSWER_R1_EXPECTED);
00577   if(response.r1 == SD_R1_NO_ERROR)
00578   {
00579     if (SD_WaitData(SD_TOKEN_START_DATA_SING
LE_BLOCK_READ) == BSP_SD_OK)
00580     {
00581       for (counter = 0; counter < 16; counte
r++)
00582       {
00583         /* Store CSD register value on CSD_T
ab */
00584         CSD_Tab[counter] = SD_IO_WriteByte(S
D_DUMMY_BYTE);
00585       }
00586
00587       /* Get CRC bytes (not really needed by
 us, but required by SD) */
00588       SD_IO_WriteByte(SD_DUMMY_BYTE);
00589       SD_IO_WriteByte(SD_DUMMY_BYTE);
00590
00591       /*********************************
*********************************
00592         CSD header decoding
00593       *********************************
*********************************/
```

```
00594
00595       /* Byte 0 */
00596       Csd->CSDStruct = (CSD_Tab[0] & 0xC0) >
> 6;
00597       Csd->Reserved1 =  CSD_Tab[0] & 0x3F;
00598
00599       /* Byte 1 */
00600       Csd->TAAC = CSD_Tab[1];
00601
00602       /* Byte 2 */
00603       Csd->NSAC = CSD_Tab[2];
00604
00605       /* Byte 3 */
00606       Csd->MaxBusClkFrec = CSD_Tab[3];
00607
00608       /* Byte 4/5 */
00609       Csd->CardComdClasses = (CSD_Tab[4] <<
4) | ((CSD_Tab[5] & 0xF0) >> 4);
00610       Csd->RdBlockLen = CSD_Tab[5] & 0x0F;
00611
00612       /* Byte 6 */
00613       Csd->PartBlockRead   = (CSD_Tab[6] & 0
x80) >> 7;
00614       Csd->WrBlockMisalign = (CSD_Tab[6] & 0
x40) >> 6;
00615       Csd->RdBlockMisalign = (CSD_Tab[6] & 0
x20) >> 5;
00616       Csd->DSRImpl         = (CSD_Tab[6] & 0
x10) >> 4;
00617
00618       /*****************************************
*************************************
00619         CSD v1/v2 decoding
00620       *****************************************
*************************************/
00621
00622       if(flag_SDHC == 0)
```

```
00623       {
00624         Csd->version.v1.Reserved1 = ((CSD_Ta
b[6] & 0x0C) >> 2);
00625
00626         Csd->version.v1.DeviceSize =  ((CSD_
Tab[6] & 0x03) << 10)
00627                                     |  (CSD_
Tab[7] << 2)
00628                                     | ((CSD_
Tab[8] & 0xC0) >> 6);
00629         Csd->version.v1.MaxRdCurrentVDDMin =
 (CSD_Tab[8] & 0x38) >> 3;
00630         Csd->version.v1.MaxRdCurrentVDDMax =
 (CSD_Tab[8] & 0x07);
00631         Csd->version.v1.MaxWrCurrentVDDMin =
 (CSD_Tab[9] & 0xE0) >> 5;
00632         Csd->version.v1.MaxWrCurrentVDDMax =
 (CSD_Tab[9] & 0x1C) >> 2;
00633         Csd->version.v1.DeviceSizeMul = ((CS
D_Tab[9] & 0x03) << 1)
00634                                       |((CS
D_Tab[10] & 0x80) >> 7);
00635       }
00636       else
00637       {
00638         Csd->version.v2.Reserved1 = ((CSD_Ta
b[6] & 0x0F) << 2) | ((CSD_Tab[7] & 0xC0) >> 6);
00639         Csd->version.v2.DeviceSize= ((CSD_Ta
b[7] & 0x3F) << 16) | (CSD_Tab[8] << 8) | CSD_Tab[
9];
00640         Csd->version.v2.Reserved2 = ((CSD_Ta
b[10] & 0x80) >> 8);
00641       }
00642
00643       Csd->EraseSingleBlockEnable = (CSD_Tab
[10] & 0x40) >> 6;
00644       Csd->EraseSectorSize   = ((CSD_Tab[10]
```

```
 & 0x3F) << 1)
00645                                |((CSD_Tab[11]
 & 0x80) >> 7);
00646       Csd->WrProtectGrSize   = (CSD_Tab[11]
& 0x7F);
00647       Csd->WrProtectGrEnable = (CSD_Tab[12]
& 0x80) >> 7;
00648       Csd->Reserved2         = (CSD_Tab[12]
& 0x60) >> 5;
00649       Csd->WrSpeedFact       = (CSD_Tab[12]
& 0x1C) >> 2;
00650       Csd->MaxWrBlockLen     = ((CSD_Tab[12]
 & 0x03) << 2)
00651                               |((CSD_Tab[13]
 & 0xC0) >> 6);
00652       Csd->WriteBlockPartial = (CSD_Tab[13]
& 0x20) >> 5;
00653       Csd->Reserved3         = (CSD_Tab[13]
& 0x1F);
00654       Csd->FileFormatGrouop  = (CSD_Tab[14]
& 0x80) >> 7;
00655       Csd->CopyFlag          = (CSD_Tab[14]
& 0x40) >> 6;
00656       Csd->PermWrProtect     = (CSD_Tab[14]
& 0x20) >> 5;
00657       Csd->TempWrProtect     = (CSD_Tab[14]
& 0x10) >> 4;
00658       Csd->FileFormat        = (CSD_Tab[14]
& 0x0C) >> 2;
00659       Csd->Reserved4         = (CSD_Tab[14]
& 0x03);
00660       Csd->crc               = (CSD_Tab[15]
& 0xFE) >> 1;
00661       Csd->Reserved5         = (CSD_Tab[15]
& 0x01);
00662
00663       retr = BSP_SD_OK;
```

```
00664     }
00665   }
00666
00667   /* Send dummy byte: 8 Clock pulses of dela
y */
00668   SD_IO_CSState(1);
00669   SD_IO_WriteByte(SD_DUMMY_BYTE);
00670
00671   /* Return the reponse */
00672   return retr;
00673 }
00674
00675 /**
00676   * @brief  Reads the SD card CID register.
00677   *         Reading the contents of the CID
register in SPI mode is a simple
00678   *         read-block transaction.
00679   * @param  Cid pointer on an CID register s
tructure
00680   * @retval SD status
00681   */
00682 uint8_t SD_GetCIDRegister(SD_CID* Cid)
00683 {
00684   uint32_t counter = 0;
00685   uint8_t retr = BSP_SD_ERROR;
00686   uint8_t CID_Tab[16];
00687   SD_CmdAnswer_typedef response;
00688
00689   /* Send CMD10 (CID register) and Wait for
response in the R1 format (0x00 is no errors) */
00690   response = SD_SendCmd(SD_CMD_SEND_CID, 0,
0xFF, SD_ANSWER_R1_EXPECTED);
00691   if(response.r1 == SD_R1_NO_ERROR)
00692   {
00693     if(SD_WaitData(SD_TOKEN_START_DATA_SINGL
E_BLOCK_READ) == BSP_SD_OK)
00694     {
```

```
00695       /* Store CID register value on CID_Tab
 */
00696       for (counter = 0; counter < 16; counte
r++)
00697       {
00698         CID_Tab[counter] = SD_IO_WriteByte(S
D_DUMMY_BYTE);
00699       }
00700
00701       /* Get CRC bytes (not really needed by
 us, but required by SD) */
00702       SD_IO_WriteByte(SD_DUMMY_BYTE);
00703       SD_IO_WriteByte(SD_DUMMY_BYTE);
00704
00705       /* Byte 0 */
00706       Cid->ManufacturerID = CID_Tab[0];
00707
00708       /* Byte 1 */
00709       Cid->OEM_AppliID = CID_Tab[1] << 8;
00710
00711       /* Byte 2 */
00712       Cid->OEM_AppliID |= CID_Tab[2];
00713
00714       /* Byte 3 */
00715       Cid->ProdName1 = CID_Tab[3] << 24;
00716
00717       /* Byte 4 */
00718       Cid->ProdName1 |= CID_Tab[4] << 16;
00719
00720       /* Byte 5 */
00721       Cid->ProdName1 |= CID_Tab[5] << 8;
00722
00723       /* Byte 6 */
00724       Cid->ProdName1 |= CID_Tab[6];
00725
00726       /* Byte 7 */
00727       Cid->ProdName2 = CID_Tab[7];
```

```
00728
00729       /* Byte 8 */
00730       Cid->ProdRev = CID_Tab[8];
00731
00732       /* Byte 9 */
00733       Cid->ProdSN = CID_Tab[9] << 24;
00734
00735       /* Byte 10 */
00736       Cid->ProdSN |= CID_Tab[10] << 16;
00737
00738       /* Byte 11 */
00739       Cid->ProdSN |= CID_Tab[11] << 8;
00740
00741       /* Byte 12 */
00742       Cid->ProdSN |= CID_Tab[12];
00743
00744       /* Byte 13 */
00745       Cid->Reserved1 |= (CID_Tab[13] & 0xF0)
 >> 4;
00746       Cid->ManufactDate = (CID_Tab[13] & 0x0
F) << 8;
00747
00748       /* Byte 14 */
00749       Cid->ManufactDate |= CID_Tab[14];
00750
00751       /* Byte 15 */
00752       Cid->CID_CRC = (CID_Tab[15] & 0xFE) >>
 1;
00753       Cid->Reserved2 = 1;
00754
00755       retr = BSP_SD_OK;
00756     }
00757   }
00758
00759   /* Send dummy byte: 8 Clock pulses of dela
y */
00760   SD_IO_CSState(1);
```

```
00761   SD_IO_WriteByte(SD_DUMMY_BYTE);
00762
00763   /* Return the reponse */
00764   return retr;
00765 }
00766
00767 /**
00768   * @brief  Send 5 bytes command to the SD c
ard and get response
00769   * @param  Cmd The user expected command to
 send to SD card.
00770   * @param  Arg The command argument.
00771   * @param  Crc The CRC.
00772   * @param  Answer SD_ANSWER_NOT_EXPECTED or
 SD_ANSWER_EXPECTED
00773   * @retval SD status
00774   */
00775 SD_CmdAnswer_typedef SD_SendCmd(uint8_t Cmd,
 uint32_t Arg, uint8_t Crc, uint8_t Answer)
00776 {
00777   uint8_t frame[SD_CMD_LENGTH], frameout[SD_
CMD_LENGTH];
00778   SD_CmdAnswer_typedef retr = {0xFF, 0xFF ,
0xFF, 0xFF, 0xFF};
00779
00780   /* R1 Lenght = NCS(0)+ 6 Bytes command + N
CR(min1 max8) + 1 Bytes answer + NEC(0) = 15bytes
*/
00781   /* R1b identical to R1 + Busy information

*/
00782   /* R2 Lenght = NCS(0)+ 6 Bytes command + N
CR(min1 max8) + 2 Bytes answer + NEC(0) = 16bytes
*/
00783
00784   /* Prepare Frame to send */
00785   frame[0] = (Cmd | 0x40);         /* Constr
```

```
uct byte 1 */
00786   frame[1] = (uint8_t)(Arg >> 24); /* Constr
uct byte 2 */
00787   frame[2] = (uint8_t)(Arg >> 16); /* Constr
uct byte 3 */
00788   frame[3] = (uint8_t)(Arg >> 8);  /* Constr
uct byte 4 */
00789   frame[4] = (uint8_t)(Arg);       /* Constr
uct byte 5 */
00790   frame[5] = (Crc | 0x01);         /* Constr
uct byte 6 */
00791
00792   /* Send the command */
00793   SD_IO_CSState(0);
00794   SD_IO_WriteReadData(frame, frameout, SD_CM
D_LENGTH); /* Send the Cmd bytes */
00795
00796   switch(Answer)
00797   {
00798   case SD_ANSWER_R1_EXPECTED :
00799     retr.r1 = SD_ReadData();
00800     break;
00801   case SD_ANSWER_R1B_EXPECTED :
00802     retr.r1 = SD_ReadData();
00803     retr.r2 = SD_IO_WriteByte(SD_DUMMY_BYTE)
;
00804     /* Set CS High */
00805     SD_IO_CSState(1);
00806     HAL_Delay(1);
00807     /* Set CS Low */
00808     SD_IO_CSState(0);
00809
00810     /* Wait IO line return 0xFF */
00811     while (SD_IO_WriteByte(SD_DUMMY_BYTE) !=
 0xFF);
00812     break;
00813   case SD_ANSWER_R2_EXPECTED :
```

```
00814     retr.r1 = SD_ReadData();
00815     retr.r2 = SD_IO_WriteByte(SD_DUMMY_BYTE)
;
00816     break;
00817   case SD_ANSWER_R3_EXPECTED :
00818   case SD_ANSWER_R7_EXPECTED :
00819     retr.r1 = SD_ReadData();
00820     retr.r2 = SD_IO_WriteByte(SD_DUMMY_BYTE)
;
00821     retr.r3 = SD_IO_WriteByte(SD_DUMMY_BYTE)
;
00822     retr.r4 = SD_IO_WriteByte(SD_DUMMY_BYTE)
;
00823     retr.r5 = SD_IO_WriteByte(SD_DUMMY_BYTE)
;
00824     break;
00825   default :
00826     break;
00827   }
00828   return retr;
00829 }
00830
00831 /**
00832   * @brief  Gets the SD card data response a
nd check the busy flag.
00833   * @retval The SD status: Read data respons
e xxx0<status>1
00834   *         - status 010: Data accecpted
00835   *         - status 101: Data rejected due
to a crc error
00836   *         - status 110: Data rejected due
to a Write error.
00837   *         - status 111: Data rejected due
to other error.
00838   */
00839 uint8_t SD_GetDataResponse(void)
00840 {
```

```
00841   uint8_t dataresponse;
00842   uint8_t rvalue = SD_DATA_OTHER_ERROR;
00843
00844   dataresponse = SD_IO_WriteByte(SD_DUMMY_BY
TE);
00845   SD_IO_WriteByte(SD_DUMMY_BYTE); /* read th
e busy response byte*/
00846
00847   /* Mask unused bits */
00848   switch (dataresponse & 0x1F)
00849   {
00850   case SD_DATA_OK:
00851     rvalue = SD_DATA_OK;
00852
00853     /* Set CS High */
00854     SD_IO_CSState(1);
00855     /* Set CS Low */
00856     SD_IO_CSState(0);
00857
00858     /* Wait IO line return 0xFF */
00859     while (SD_IO_WriteByte(SD_DUMMY_BYTE) !=
 0xFF);
00860     break;
00861   case SD_DATA_CRC_ERROR:
00862     rvalue =  SD_DATA_CRC_ERROR;
00863     break;
00864   case SD_DATA_WRITE_ERROR:
00865     rvalue = SD_DATA_WRITE_ERROR;
00866     break;
00867   default:
00868     break;
00869   }
00870
00871   /* Return response */
00872   return rvalue;
00873 }
00874
```

```
00875
00876 /**
00877   * @brief  Put the SD in Idle state.
00878   * @retval SD status
00879   */
00880 uint8_t SD_GoIdleState(void)
00881 {
00882   SD_CmdAnswer_typedef response;
00883   __IO uint8_t counter = 0;
00884   /* Send CMD0 (SD_CMD_GO_IDLE_STATE) to put
 SD in SPI mode and
00885      wait for In Idle State Response (R1 For
mat) equal to 0x01 */
00886   do{
00887     counter++;
00888     response = SD_SendCmd(SD_CMD_GO_IDLE_STA
TE, 0, 0x95, SD_ANSWER_R1_EXPECTED);
00889     SD_IO_CSState(1);
00890     SD_IO_WriteByte(SD_DUMMY_BYTE);
00891     if(counter >= SD_MAX_TRY)
00892     {
00893       return BSP_SD_ERROR;
00894     }
00895   }
00896   while(response.r1 != SD_R1_IN_IDLE_STATE);
00897
00898
00899   /* Send CMD8 (SD_CMD_SEND_IF_COND) to chec
k the power supply status
00900      and wait until response (R7 Format) equ
al to 0xAA and */
00901   response = SD_SendCmd(SD_CMD_SEND_IF_COND,
 0x1AA, 0x87, SD_ANSWER_R7_EXPECTED);
00902   SD_IO_CSState(1);
00903   SD_IO_WriteByte(SD_DUMMY_BYTE);
00904   if((response.r1  & SD_R1_ILLEGAL_COMMAND)
== SD_R1_ILLEGAL_COMMAND)
```

```
00905   {
00906     /* initialise card V1 */
00907     do
00908     {
00909       /* initialise card V1 */
00910       /* Send CMD55 (SD_CMD_APP_CMD) before
any ACMD command: R1 response (0x00: no errors) */

00911       response = SD_SendCmd(SD_CMD_APP_CMD,
0x00000000, 0xFF, SD_ANSWER_R1_EXPECTED);
00912       SD_IO_CSState(1);
00913       SD_IO_WriteByte(SD_DUMMY_BYTE);
00914
00915       /* Send ACMD41 (SD_CMD_SD_APP_OP_COND)
 to initialize SDHC or SDXC cards: R1 response (0x
00: no errors) */
00916       response = SD_SendCmd(SD_CMD_SD_APP_OP
_COND, 0x00000000, 0xFF, SD_ANSWER_R1_EXPECTED);
00917       SD_IO_CSState(1);
00918       SD_IO_WriteByte(SD_DUMMY_BYTE);
00919     }
00920     while(response.r1 == SD_R1_IN_IDLE_STATE
);
00921     flag_SDHC = 0;
00922   }
00923   else if(response.r1 == SD_R1_IN_IDLE_STATE
)
00924   {
00925       /* initialise card V2 */
00926     do {
00927
00928       /* Send CMD55 (SD_CMD_APP_CMD) before
any ACMD command: R1 response (0x00: no errors) */

00929       response = SD_SendCmd(SD_CMD_APP_CMD,
0, 0xFF, SD_ANSWER_R1_EXPECTED);
00930       SD_IO_CSState(1);
```

```
00931       SD_IO_WriteByte(SD_DUMMY_BYTE);
00932
00933       /* Send ACMD41 (SD_CMD_SD_APP_OP_COND)
 to initialize SDHC or SDXC cards: R1 response (0x
00: no errors) */
00934       response = SD_SendCmd(SD_CMD_SD_APP_OP
_COND, 0x40000000, 0xFF, SD_ANSWER_R1_EXPECTED);
00935       SD_IO_CSState(1);
00936       SD_IO_WriteByte(SD_DUMMY_BYTE);
00937     }
00938     while(response.r1 == SD_R1_IN_IDLE_STATE
);
00939
00940     if((response.r1 & SD_R1_ILLEGAL_COMMAND)
 == SD_R1_ILLEGAL_COMMAND)
00941     {
00942       do {
00943         /* Send CMD55 (SD_CMD_APP_CMD) befor
e any ACMD command: R1 response (0x00: no errors)
*/
00944         response = SD_SendCmd(SD_CMD_APP_CMD
, 0, 0xFF, SD_ANSWER_R1_EXPECTED);
00945         SD_IO_CSState(1);
00946         SD_IO_WriteByte(SD_DUMMY_BYTE);
00947         if(response.r1 != SD_R1_IN_IDLE_STATE
)
00948         {
00949           return BSP_SD_ERROR;
00950         }
00951         /* Send ACMD41 (SD_CMD_SD_APP_OP_CON
D) to initialize SDHC or SDXC cards: R1 response (
0x00: no errors) */
00952         response = SD_SendCmd(SD_CMD_SD_APP_
OP_COND, 0x00000000, 0xFF, SD_ANSWER_R1_EXPECTED);
00953         SD_IO_CSState(1);
00954         SD_IO_WriteByte(SD_DUMMY_BYTE);
00955       }
```

```
00956       while(response.r1 == SD_R1_IN_IDLE_STA
TE);
00957     }
00958
00959     /* Send CMD58 (SD_CMD_READ_OCR) to initi
alize SDHC or SDXC cards: R3 response (0x00: no er
rors) */
00960     response = SD_SendCmd(SD_CMD_READ_OCR, 0
x00000000, 0xFF, SD_ANSWER_R3_EXPECTED);
00961     SD_IO_CSState(1);
00962     SD_IO_WriteByte(SD_DUMMY_BYTE);
00963     if(response.r1 != SD_R1_NO_ERROR)
00964     {
00965       return BSP_SD_ERROR;
00966     }
00967     flag_SDHC = (response.r2 & 0x40) >> 6;
00968   }
00969   else
00970   {
00971     return BSP_SD_ERROR;
00972   }
00973
00974   return BSP_SD_OK;
00975 }
00976
00977 /**
00978   * @brief  Waits a data until a value diffe
rent from SD_DUMMY_BITE
00979   * @retval the value read
00980   */
00981 uint8_t SD_ReadData(void)
00982 {
00983   uint8_t timeout = 0x08;
00984   uint8_t readvalue;
00985
00986   /* Check if response is got or a timeout i
s happen */
```

```
00987   do {
00988     readvalue = SD_IO_WriteByte(SD_DUMMY_BYTE
);
00989     timeout--;
00990
00991   }while ((readvalue == SD_DUMMY_BYTE) && ti
meout);
00992
00993   /* Right response got */
00994   return readvalue;
00995 }
00996
00997 /**
00998   * @brief  Waits a data from the SD card
00999   * @param  data  Expected data from the SD
card
01000   * @retval BSP_SD_OK or BSP_SD_TIMEOUT
01001   */
01002 uint8_t SD_WaitData(uint8_t data)
01003 {
01004   uint16_t timeout = 0xFFFF;
01005   uint8_t readvalue;
01006
01007   /* Check if response is got or a timeout i
s happen */
01008
01009   do {
01010     readvalue = SD_IO_WriteByte(SD_DUMMY_BYTE
);
01011     timeout--;
01012   }while ((readvalue != data) && timeout);
01013
01014   if (timeout == 0)
01015   {
01016     /* After time out */
01017     return BSP_SD_TIMEOUT;
01018   }
```

```
01019
01020   /* Right response got */
01021   return BSP_SD_OK;
01022 }
01023
01024 /**
01025   * @}
01026   */
01027
01028 /**
01029   * @}
01030   */
01031
01032 /**
01033   * @}
01034   */
01035
01036 /**
01037   * @}
01038   */
01039
01040 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32072B_EVAL BSP User Manual

Exported Types

STM32072B_EVAL SD

Data Structures

struct	**struct_v1**
struct	**struct_v2**
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...

Enumerations

enum	{ **BSP_SD_OK** = 0x00, **MSD_OK** = 0x00, **BSP_SD_ERROR** = 0x01, **MSD_ERROR** = 0x01, **BSP_SD_TIMEOUT** } SD status structure definition. More...

Enumeration Type Documentation

anonymous enum

SD status structure definition.

Enumerator:

BSP_SD_OK

MSD_OK

BSP_SD_ERROR

MSD_ERROR

BSP_SD_TIMEOUT

Definition at line **67** of file **stm32072b_eval_sd.h**.

STM32072B_EVAL BSP User Manual

stm32072b_eval_sd.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32072b_eval_sd.h
00004   * @author  MCD Application Team
00005   * @brief   This file contains the common defines and functions prototypes for
00006   *          the stm32072b_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 __STM32072B_EVAL_SD_H
00039 #define __STM32072B_EVAL_SD_H
00040
00041 #ifdef __cplusplus
00042  extern "C" {
00043 #endif
00044
00045 /* Includes ------------------------------
-------------------------------------*/
00046 #include "stm32072b_eval.h"
00047
00048 /** @addtogroup BSP
00049   * @{
00050   */
00051
00052 /** @addtogroup STM32072B_EVAL
00053   * @{
00054   */
00055
00056 /** @defgroup STM32072B_EVAL_SD STM32072B_EV
AL SD
00057   * @{
00058   */
00059
00060 /** @defgroup STM32072B_EVAL_SD_Exported_Typ
es Exported Types
00061   * @{
00062   */
00063
00064 /**
00065   * @brief  SD status structure definition
00066   */
00067 enum {
```

```
00068       BSP_SD_OK = 0x00,
00069       MSD_OK = 0x00,
00070       BSP_SD_ERROR = 0x01,
00071       MSD_ERROR    = 0x01,
00072       BSP_SD_TIMEOUT
00073 };
00074
00075 typedef struct
00076 {
00077   uint8_t  Reserved1:2;                 /* Res
erved */
00078   uint16_t DeviceSize:12;               /* Dev
ice Size */
00079   uint8_t  MaxRdCurrentVDDMin:3;        /* Max
. read current @ VDD min */
00080   uint8_t  MaxRdCurrentVDDMax:3;        /* Max
. read current @ VDD max */
00081   uint8_t  MaxWrCurrentVDDMin:3;        /* Max
. write current @ VDD min */
00082   uint8_t  MaxWrCurrentVDDMax:3;        /* Max
. write current @ VDD max */
00083   uint8_t  DeviceSizeMul:3;             /* Dev
ice size multiplier */
00084 } struct_v1;
00085
00086
00087 typedef struct
00088 {
00089   uint8_t  Reserved1:6;                 /* Res
erved */
00090   uint32_t DeviceSize:22;               /* Dev
ice Size */
00091   uint8_t  Reserved2:1;                 /* Res
erved */
00092 } struct_v2;
00093
00094 /**
```

```
00095   * @brief  Card Specific Data: CSD Register
00096   */
00097 typedef struct
00098 {
00099   /* Header part */
00100   uint8_t  CSDStruct:2;            /* CSD st
ructure */
00101   uint8_t  Reserved1:6;            /* Reserv
ed */
00102   uint8_t  TAAC:8;                 /* Data r
ead access-time 1 */
00103   uint8_t  NSAC:8;                 /* Data r
ead access-time 2 in CLK cycles */
00104   uint8_t  MaxBusClkFrec:8;        /* Max. b
us clock frequency */
00105   uint16_t CardComdClasses:12;      /* Card
command classes */
00106   uint8_t  RdBlockLen:4;           /* Max. r
ead data block length */
00107   uint8_t  PartBlockRead:1;        /* Partia
l blocks for read allowed */
00108   uint8_t  WrBlockMisalign:1;      /* Write
block misalignment */
00109   uint8_t  RdBlockMisalign:1;      /* Read b
lock misalignment */
00110   uint8_t  DSRImpl:1;              /* DSR im
plemented */
00111
00112   /* v1 or v2 struct */
00113   union csd_version {
00114     struct_v1 v1;
00115     struct_v2 v2;
00116   } version;
00117
00118   uint8_t  EraseSingleBlockEnable:1;  /* Era
se single block enable */
00119   uint8_t  EraseSectorSize:7;         /* Era
```

```
se group size multiplier */
00120   uint8_t  WrProtectGrSize:7;          /* Wri
te protect group size */
00121   uint8_t  WrProtectGrEnable:1;        /* Wri
te protect group enable */
00122   uint8_t  Reserved2:2;                /* Res
erved */
00123   uint8_t  WrSpeedFact:3;              /* Wri
te speed factor */
00124   uint8_t  MaxWrBlockLen:4;            /* Max
. write data block length */
00125   uint8_t  WriteBlockPartial:1;        /* Par
tial blocks for write allowed */
00126   uint8_t  Reserved3:5;                /* Res
erved */
00127   uint8_t  FileFormatGrouop:1;         /* Fil
e format group */
00128   uint8_t  CopyFlag:1;                 /* Cop
y flag (OTP) */
00129   uint8_t  PermWrProtect:1;            /* Per
manent write protection */
00130   uint8_t  TempWrProtect:1;            /* Tem
porary write protection */
00131   uint8_t  FileFormat:2;               /* Fil
e Format */
00132   uint8_t  Reserved4:2;                /* Res
erved */
00133   uint8_t  crc:7;                      /* Res
erved */
00134   uint8_t  Reserved5:1;                /* alw
ays 1*/
00135
00136 } SD_CSD;
00137
00138 /**
00139   * @brief  Card Identification Data: CID Re
gister
```

```
00140   */
00141 typedef struct
00142 {
00143   __IO uint8_t  ManufacturerID;       /* Man
ufacturerID */
00144   __IO uint16_t OEM_AppliID;          /* OEM
/Application ID */
00145   __IO uint32_t ProdName1;            /* Pro
duct Name part1 */
00146   __IO uint8_t  ProdName2;            /* Pro
duct Name part2*/
00147   __IO uint8_t  ProdRev;              /* Pro
duct Revision */
00148   __IO uint32_t ProdSN;               /* Pro
duct Serial Number */
00149   __IO uint8_t  Reserved1;            /* Res
erved1 */
00150   __IO uint16_t ManufactDate;         /* Man
ufacturing Date */
00151   __IO uint8_t  CID_CRC;              /* CID
 CRC */
00152   __IO uint8_t  Reserved2;            /* alw
ays 1 */
00153 } SD_CID;
00154
00155 /**
00156   * @brief SD Card information
00157   */
00158 typedef struct
00159 {
00160   SD_CSD Csd;
00161   SD_CID Cid;
00162   uint32_t CardCapacity;  /* Card Capacity */

00163   uint32_t CardBlockSize; /* Card Block Size
 */
00164 } SD_CardInfo;
```

```
00165
00166 /**
00167   * @}
00168   */
00169
00170 /** @defgroup STM32072B_EVAL_SPI_SD_Exported
_Constants Exported Constants
00171   * @{
00172   */
00173
00174 /**
00175   * @brief  Block Size
00176   */
00177 #define SD_BLOCK_SIZE    0x200
00178
00179 /**
00180   * @brief  SD detection on its memory slot
00181   */
00182 #define SD_PRESENT               ((uint8_t)0
x01)
00183 #define SD_NOT_PRESENT           ((uint8_t)0
x00)
00184
00185 /**
00186   * @}
00187   */
00188
00189 /** @defgroup STM32091C_EVAL_SD_Exported_Mac
ro Exported Macro
00190   * @{
00191   */
00192
00193 /**
00194   * @}
00195   */
00196
00197 /** @defgroup STM32072B_EVAL_SD_Exported_Fun
```

```
ctions Exported Functions
00198   * @{
00199   */
00200 uint8_t BSP_SD_Init(void);
00201 uint8_t BSP_SD_IsDetected(void);
00202 uint8_t BSP_SD_ReadBlocks(uint32_t *pData, u
int32_t ReadAddr, uint16_t BlockSize, uint32_t Num
berOfBlocks);
00203 uint8_t BSP_SD_WriteBlocks(uint32_t *pData,
uint32_t WriteAddr, uint16_t BlockSize, uint32_t N
umberOfBlocks);
00204 uint8_t BSP_SD_Erase(uint32_t StartAddr, uin
t32_t EndAddr);
00205 uint8_t BSP_SD_GetStatus(void);
00206 uint8_t BSP_SD_GetCardInfo(SD_CardInfo *pCar
dInfo);
00207
00208 /* Link functions for SD Card peripheral */
00209 void    SD_IO_Init(void);
00210 void    SD_IO_CSState(uint8_t state);
00211 void    SD_IO_WriteReadData(const uint8_t *D
ataIn, uint8_t *DataOut, uint16_t DataLength);
00212 uint8_t SD_IO_WriteByte(uint8_t Data);
00213
00214 /**
00215   * @}
00216   */
00217
00218 /** @defgroup STM32072B_EVAL_SD_LINK_Operati
ons_Functions LINK Operations Functions
00219   * @{
00220   */
00221
00222 /**
00223   * @}
00224   */
00225
```

```
00226 /**
00227   * @}
00228   */
00229
00230 /**
00231   * @}
00232   */
00233
00234 /**
00235   * @}
00236   */
00237
00238 #ifdef __cplusplus
00239 }
00240 #endif
00241
00242 #endif /* __STM32072B_EVAL_SD_H */
00243
00244 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32072B_EVAL BSP User Manual

Data Fields

SD_CSD::csd_version Union Reference

`#include <stm32072b_eval_sd.h>`

Data Fields

struct_v1	**v1**
struct_v2	**v2**

Detailed Description

Definition at line **113** of file **stm32072b_eval_sd.h**.

Field Documentation

struct_v1 SD_CSD::csd_version::v1

Definition at line **114** of file **stm32072b_eval_sd.h**.

Referenced by **BSP_SD_GetCardInfo()**, and **SD_GetCSDRegister()**.

struct_v2 SD_CSD::csd_version::v2

Definition at line **115** of file **stm32072b_eval_sd.h**.

Referenced by **BSP_SD_GetCardInfo()**, and **SD_GetCSDRegister()**.

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

- **stm32072b_eval_sd.h**

STM32072B_EVAL BSP User Manual

STM32072B_EVAL SD

STM32072B_EVAL

Modules

STM32072B_EVAL BSP User Manual

Private Constants

STM32072B_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	**__STM32072B_EVAL_BSP_VERSION_MAIN** (0x02) STM32072B EVAL BSP Driver version number V2.1.8.
#define	**__STM32072B_EVAL_BSP_VERSION_SUB1** (0x01)
#define	**__STM32072B_EVAL_BSP_VERSION_SUB2** (0x08)
#define	**__STM32072B_EVAL_BSP_VERSION_RC** (0x00)
#define	**__STM32072B_EVAL_BSP_VERSION**

Define Documentation

#define __STM32072B_EVAL_BSP_VERSION

Value:

```
((__STM32072B_EVAL_BSP_VERSION_MAIN << 24)\
                                                |(__
STM32072B_EVAL_BSP_VERSION_SUB1 << 16)\
                                                |(__
STM32072B_EVAL_BSP_VERSION_SUB2 << 8 )\
                                                |(__
STM32072B_EVAL_BSP_VERSION_RC))
```

Definition at line **76** of file **stm32072b_eval.c**.

Referenced by **BSP_GetVersion()**.

#define __STM32072B_EVAL_BSP_VERSION_MAIN (0x02)

STM32072B EVAL BSP Driver version number V2.1.8.

[31:24] main version

Definition at line **72** of file **stm32072b_eval.c**.

#define __STM32072B_EVAL_BSP_VERSION_RC (0x00)

[7:0] release candidate

Definition at line **75** of file **stm32072b_eval.c**.

#define __STM32072B_EVAL_BSP_VERSION_SUB1 (0x01)

[23:16] sub1 version

Definition at line **73** of file **stm32072b_eval.c**.

#define __STM32072B_EVAL_BSP_VERSION_SUB2 (0x08)

[15:8] sub2 version

Definition at line **74** of file **stm32072b_eval.c**.

#define LCD_READ_REG 0x03

Definition at line **63** of file **stm32072b_eval.c**.

Referenced by **LCD_IO_ReadData()**.

#define LCD_WRITE_REG 0x02

Definition at line **62** of file **stm32072b_eval.c**.

Referenced by **LCD_IO_WriteMultipleData()**.

#define READ_STATUS 0x01

Definition at line **61** of file **stm32072b_eval.c**.

#define SD_DUMMY_BYTE 0xFF

Definition at line **66** of file **stm32072b_eval.c**.

Referenced by **SD_IO_Init()**.

#define SD_NO_RESPONSE_EXPECTED 0x80

Definition at line 67 of file stm32072b_eval.c.

#define SET_INDEX 0x00

Definition at line 60 of file stm32072b_eval.c.

Referenced by LCD_IO_WriteReg().

#define START_BYTE 0x70

Definition at line 59 of file stm32072b_eval.c.

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

STM32072B_EVAL BSP User Manual

Private Macros

STM32072B_EVAL LCD

Defines

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

Define Documentation

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

Definition at line **100** of file **stm32072b_eval_lcd.c**.

Referenced by **BSP_LCD_DrawLine()**.

STM32072B_EVAL BSP User Manual

Private Variables

STM32072B_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 114 of file stm32072b_eval_lcd.c.

Referenced by LCD_DrawChar().

LCD_DrawPropTypeDef DrawProp

Definition at line 109 of file stm32072b_eval_lcd.c.

LCD_DrvTypeDef* lcd_drv [static]

Definition at line 111 of file stm32072b_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().

STM32072B_EVAL BSP User Manual

Exported Functions

STM32072B_EVAL Common

Functions

uint32_t	**BSP_GetVersion** (void) This method returns the STM32F072B 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** Mode) Configures Tamper Button GPIO or EXTI Line.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
uint8_t	**BSP_JOY_Init** (**JOYMode_TypeDef** Joy_Mode) Configures joystick GPIO and EXTI modes.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the current joystick status.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) Configures COM port.

Function Documentation

void BSP_COM_Init (COM_TypeDef COM,
UART_HandleTypeDef * huart
)

Configures COM port.

Parameters:

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

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

Return values:

None

Definition at line **481** of file **stm32072b_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 STM32F072B EVAL BSP Driver revision.

Return values:

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

Definition at line **264** of file **stm32072b_eval.c**.

References __STM32072B_EVAL_BSP_VERSION.

JOYState_TypeDef BSP_JOY_GetState (void)

Returns the current joystick status.

Return values:

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

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

Definition at line 454 of file stm32072b_eval.c.

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

uint8_t BSP_JOY_Init (JOYMode_TypeDef Joy_Mode)

Configures joystick GPIO and EXTI modes.

Parameters:

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

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

Return values:

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

Definition at line **406** of file **stm32072b_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 **279** of file **stm32072b_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 **322** of file **stm32072b_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 **307** of file **stm32072b_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 **337** of file **stm32072b_eval.c**.

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

uint32_t BSP_PB_GetState (Button_TypeDef Button)

Returns the selected button state.

Parameters:

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

- BUTTON_TAMPER: Tamper Push Button

Return values:

The Button GPIO pin value

Definition at line **392** of file **stm32072b_eval.c**.

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

void BSP_PB_Init (Button_TypeDef Button,
ButtonMode_TypeDef Mode
)

Configures Tamper Button GPIO or EXTI Line.

Parameters:

Button Button to be configured This parameter can be one of the following values:
- BUTTON_TAMPER: Tamper Push Button

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

Return values:

None

Definition at line **354** of file **stm32072b_eval.c**.

References **BUTTON_IRQn**, **BUTTON_MODE_EXTI**, **BUTTON_MODE_GPIO**, **BUTTON_PIN**, **BUTTON_PORT**, and **TAMPERx_GPIO_CLK_ENABLE**.

STM32072B_EVAL BSP User Manual

Exported Functions

STM32072B_EVAL EEPROM

Functions

uint32_t	**BSP_EEPROM_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
uint32_t	**BSP_EEPROM_ReadBuffer** (uint8_t *pBuffer, uint16_t ReadAddr, 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.

Function Documentation

uint32_t BSP_EEPROM_Init (void)

Initializes peripherals used by the I2C EEPROM driver.

Note:
There are 2 different versions of M24LR64 (A01 & A02). Then try to connect on 1st one (EEPROM_I2C_ADDRESS_A01) and if problem, check the 2nd one (EEPROM_I2C_ADDRESS_A02)

Return values:

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

Definition at line **145** of file **stm32072b_eval_eeprom.c**.

References **EEPROM_FAIL**, **EEPROM_I2C_Drv**, 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 **173** of file **stm32072b_eval_eeprom.c**.

References **EEPROM_FAIL**, and **EEPROM_DrvTypeDef::ReadBuffer**.

void BSP_EEPROM_TIMEOUT_UserCallback (void)

Basic management of the timeout situation.

Return values:

None.

Definition at line **347** of file **stm32072b_eval_eeprom.c**.

Referenced by **EEPROM_I2C_WaitEepromStandbyState()**.

uint32_t BSP_EEPROM_WriteBuffer (uint8_t * pBuffer, uint16_t WriteAddr, uint32_t NumByteToWrite)

Writes buffer of data to the EEPROM device selected.

Parameters:

pBuffer pointer to the buffer containing the data to

	be written to the EEPROM.
WriteAddr	EEPROM's internal address to write to.
NumByteToWrite	number of bytes to write to the EEPROM.

Return values:

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

< If NumByteToWrite < EEPROM_PAGESIZE

< If NumByteToWrite < EEPROM_PAGESIZE

< If the number of data to be written is more than the remaining space in the current page:

< Write the data contained in same page

< Write the remaining data in the following page

Definition at line **194** of file **stm32072b_eval_eeprom.c**.

References **EEPROM_FAIL**, **EEPROM_OK**, **EEPROMPageSize**, and **EEPROM_DrvTypeDef::WritePage**.

STM32072B_EVAL BSP User Manual

Defines

Exported Constants

STM32072B_EVAL EEPROM

Defines

#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_M24LR64** 4 /* RF EEPROM ANT7-M24LR-A used */
#define	**EEPROM_OK** 0
#define	**EEPROM_FAIL** 1
#define	**EEPROM_TIMEOUT** 2
#define	**BSP_EEPROM_M24LR64** 1 /* RF I2C EEPROM M24LR64 */
#define	**EEPROM_MAX_TRIALS** 300

Define Documentation

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

Definition at line 88 of file stm32072b_eval_eeprom.h.

#define EEPROM_ADDRESS_M24LR64_A01 0xA0 /* RF EEPROM

Definition at line 77 of file stm32072b_eval_eeprom.h.

Referenced by EEPROM_I2C_Init().

#define EEPROM_ADDRESS_M24LR64_A02 0xA6 /* RF EEPROM

Definition at line 78 of file stm32072b_eval_eeprom.h.

Referenced by EEPROM_I2C_Init().

#define EEPROM_FAIL 1

Definition at line 84 of file stm32072b_eval_eeprom.h.

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

#define EEPROM_MAX_TRIALS 300

Definition at line 91 of file stm32072b_eval_eeprom.h.

Referenced by EEPROM_I2C_Init(), and EEPROM_I2C_WaitEepromStandbyState().

#define EEPROM_OK 0

Definition at line 83 of file stm32072b_eval_eeprom.h.

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

#define EEPROM_PAGESIZE_M24LR64 4 /* RF EEPROM ANT7-M2

Definition at line 80 of file stm32072b_eval_eeprom.h.

Referenced by EEPROM_I2C_Init().

#define EEPROM_TIMEOUT 2

Definition at line 85 of file stm32072b_eval_eeprom.h.

Referenced by EEPROM_I2C_WaitEepromStandbyState().

STM32072B_EVAL BSP User Manual

Exported Functions

STM32072B_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 **251** of file **stm32072b_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 **276** of file **stm32072b_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 296 of file stm32072b_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 801 of file stm32072b_eval_lcd.c.

References **lcd_drv**.

void BSP_LCD_DisplayOn (void)

Enables the display.

Return values:

None

Definition at line **792** of file **stm32072b_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 314 of file stm32072b_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 373 of file stm32072b_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 **659** of file **stm32072b_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 **547** of file **stm32072b_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 **626** of file **stm32072b_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 **403** of file **stm32072b_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 **454** of file **stm32072b_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 **597** of file **stm32072b_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 **529** of file **stm32072b_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 **427** of file **stm32072b_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 **712** of file **stm32072b_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 **762** of file **stm32072b_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 **695** of file **stm32072b_eval_lcd.c**.

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

Referenced by **BSP_LCD_ClearStringLine()**.

uint16_t BSP_LCD_GetBackColor (void)

Gets the LCD background color.

Return values:

Used background color

Definition at line **202** of file **stm32072b_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**.

sFONT * BSP_LCD_GetFont (void)

Gets the LCD text font.

Return values:

Used font

Definition at line **241** of file **stm32072b_eval_lcd.c**.

References **LCD_DrawPropTypeDef::pFont**.

uint16_t BSP_LCD_GetTextColor (void)

Gets the LCD text color.

Return values:

Used text color.

Definition at line **193** of file **stm32072b_eval_lcd.c**.

References **LCD_DrawPropTypeDef::TextColor**.

uint32_t BSP_LCD_GetXSize (void)

Gets the LCD X size.

Return values:

Used LCD X size

Definition at line **175** of file **stm32072b_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 **184** of file **stm32072b_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 **138** of file **stm32072b_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 384 of file stm32072b_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 222 of file stm32072b_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 **232** of file **stm32072b_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 **212** of file **stm32072b_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 Jul 5 2017 08:56:11 for STM32072B_EVAL BSP
User Manual by doxygen 1.7.6.1

STM32072B_EVAL BSP User Manual

Functions

Exported Functions

STM32072B_EVAL SD

Functions

uint8_t	**BSP_SD_Init** (void) Initializes the SD/SD communication.
uint8_t	**BSP_SD_IsDetected** (void) Detects if SD card is correctly plugged in the memory slot or not.
uint8_t	**BSP_SD_GetCardInfo** (**SD_CardInfo** *pCardInfo) Returns information about specific card.
uint8_t	**BSP_SD_ReadBlocks** (uint32_t *pData, uint32_t ReadAddr, uint16_t BlockSize, uint32_t NumberOfBlocks) Reads block(s) from a specified address in the SD card, in polling mode.
uint8_t	**BSP_SD_WriteBlocks** (uint32_t *pData, uint32_t WriteAddr, uint16_t BlockSize, uint32_t NumberOfBlocks) Writes block(s) to a specified address in the 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.
void	**SD_IO_Init** (void) Initializes the SD Card and put it into StandBy State (Ready for data transfer).
void	**SD_IO_CSState** (uint8_t state)
void	**SD_IO_WriteReadData** (const uint8_t *DataIn, uint8_t *DataOut, uint16_t DataLength) Write a byte on the SD.
uint8_t	**SD_IO_WriteByte** (uint8_t Data) Writes a byte on the SD.

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 **508** of file **stm32072b_eval_sd.c**.

References **BSP_SD_ERROR**, **BSP_SD_OK**, **SD_CmdAnswer_typedef::r1**, **SD_ANSWER_R1_EXPECTED**, **SD_ANSWER_R1B_EXPECTED**, **SD_CMD_ERASE**, **SD_CMD_SD_ERASE_GRP_END**, **SD_CMD_SD_ERASE_GRP_START**, **SD_DUMMY_BYTE**, **SD_IO_CSState()**, **SD_IO_WriteByte()**, **SD_R1_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 **322** of file **stm32072b_eval_sd.c**.

References **SD_CardInfo::CardBlockSize**, **SD_CardInfo::CardCapacity**, **SD_CardInfo::Cid**, **SD_CardInfo::Csd**, **struct_v1::DeviceSize**, **struct_v2::DeviceSize**, **struct_v1::DeviceSizeMul**, **SD_CSD::RdBlockLen**, **SD_GetCIDRegister()**, **SD_GetCSDRegister()**, **SD_CSD::csd_version::v1**, **SD_CSD::csd_version::v2**, and **SD_CSD::version**.

uint8_t BSP_SD_GetStatus (void)

Returns the SD status.

Return values:

The SD status.

Definition at line **543** of file **stm32072b_eval_sd.c**.

References **BSP_SD_ERROR**, **BSP_SD_OK**, **SD_CmdAnswer_typedef::r1**, **SD_CmdAnswer_typedef::r2**, **SD_ANSWER_R2_EXPECTED**, **SD_CMD_SEND_STATUS**, **SD_DUMMY_BYTE**, **SD_IO_CSState()**, **SD_IO_WriteByte()**, **SD_R1_NO_ERROR**, **SD_R2_NO_ERROR**, and **SD_SendCmd()**.

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 **277** of file **stm32072b_eval_sd.c**.

References **BSP_SD_IsDetected()**, **MSD_ERROR**, **SD_GoIdleState()**, **SD_IO_Init()**, **SD_NOT_PRESENT**, and **SD_PRESENT**.

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 **301** of file **stm32072b_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 * pData,
uint32_t ReadAddr,
uint16_t BlockSize,
uint32_t NumberOfBlocks
)

Reads block(s) from a specified address in the SD card, in polling mode.

Parameters:

pData Pointer to the buffer that will contain the data to transmit

ReadAddr	Address from where data is to be read
BlockSize	SD card data block size, that should be 512
NumberOfBlocks	Number of SD blocks to read

Return values:

SD status

Definition at line **352** of file **stm32072b_eval_sd.c**.

References **BSP_SD_ERROR**, **BSP_SD_OK**, **SD_CmdAnswer_typedef::r1**, **SD_ANSWER_R1_EXPECTED**, **SD_CMD_READ_SINGLE_BLOCK**, **SD_CMD_SET_BLOCKLEN**, **SD_DUMMY_BYTE**, **SD_IO_CSState()**, **SD_IO_WriteByte()**, **SD_IO_WriteReadData()**, **SD_R1_NO_ERROR**, **SD_SendCmd()**, **SD_TOKEN_START_DATA_SINGLE_BLOCK_READ**, and **SD_WaitData()**.

```
uint8_t BSP_SD_WriteBlocks ( uint32_t * pData,
                             uint32_t   WriteAddr,
                             uint16_t   BlockSize,
                             uint32_t   NumberOfBlocks
                           )
```

Writes block(s) to a specified address in the SD card, in polling mode.

Parameters:

pData	Pointer to the buffer that will contain the data to transmit
WriteAddr	Address from where data is to be written
BlockSize	SD card data block size, that should be 512
NumberOfBlocks	Number of SD blocks to write

Return values:

SD status

Definition at line 429 of file stm32072b_eval_sd.c.

References BSP_SD_ERROR, BSP_SD_OK, SD_CmdAnswer_typedef::r1, SD_ANSWER_R1_EXPECTED, SD_CMD_SET_BLOCKLEN, SD_CMD_WRITE_SINGLE_BLOCK, SD_DATA_OK, SD_DUMMY_BYTE, SD_GetDataResponse(), SD_IO_CSState(), SD_IO_WriteByte(), SD_IO_WriteReadData(), SD_R1_NO_ERROR, SD_SendCmd(), and SD_TOKEN_START_DATA_SINGLE_BLOCK_WRITE.

void SD_IO_CSState (uint8_t state)

Definition at line 1160 of file stm32072b_eval.c.

References SD_CS_HIGH, and SD_CS_LOW.

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

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 1109 of file stm32072b_eval.c.

References LCD_CS_HIGH, LCD_NCS_GPIO_PORT, LCD_NCS_PIN, 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_WriteByte (uint8_t Data)

Writes a byte on the SD.

Parameters:

Data byte to send.

Return values:

None

Definition at line **1190** of file **stm32072b_eval.c**.

References **SPIx_WriteReadData()**.

Referenced by **BSP_SD_Erase()**, **BSP_SD_GetStatus()**, **BSP_SD_ReadBlocks()**, **BSP_SD_WriteBlocks()**, **SD_GetCIDRegister()**, **SD_GetCSDRegister()**, **SD_GetDataResponse()**, **SD_GoIdleState()**, **SD_IO_Init()**, **SD_ReadData()**, **SD_SendCmd()**, and **SD_WaitData()**.

void SD_IO_WriteReadData (const uint8_t * DataIn,
uint8_t * DataOut,
uint16_t DataLength
)

Write a byte on the SD.

Parameters:

DataIn	byte to send.
DataOut	read byte.
DataLength	data length.

Return values:

None

Definition at line **1179** of file **stm32072b_eval.c**.

References **SPIx_WriteReadData()**.

Referenced by **BSP_SD_ReadBlocks()**, **BSP_SD_WriteBlocks()**, and **SD_SendCmd()**.

STM32072B_EVAL BSP User Manual

Exported Functions

STM32072B_EVAL TSENSOR

Functions

uint32_t	**BSP_TSENSOR_Init** (void) Initializes peripherals used by the I2C Temperature Sensor driver.
uint8_t	**BSP_TSENSOR_ReadStatus** (void) Returns the Temperature Sensor status.
uint16_t	**BSP_TSENSOR_ReadTemp** (void) Read Temperature register of STLM75.

Function Documentation

uint32_t BSP_TSENSOR_Init (void)

Initializes peripherals used by the I2C Temperature Sensor driver.

Return values:

TSENSOR status

Definition at line **95** of file **stm32072b_eval_tsensor.c**.

References **tsensor_drv**, **TSENSOR_ERROR**, **TSENSOR_I2C_ADDRESS_A01**, **TSENSOR_I2C_ADDRESS_A02**, **TSENSOR_MAX_TRIALS**, **TSENSOR_OK**, and **TSENSORAddress**.

uint8_t BSP_TSENSOR_ReadStatus (void)

Returns the Temperature Sensor status.

Return values:

The Temperature Sensor status.

Definition at line **147** of file **stm32072b_eval_tsensor.c**.

References **tsensor_drv**, and **TSENSORAddress**.

uint16_t BSP_TSENSOR_ReadTemp (void)

Read Temperature register of STLM75.

Return values:

STLM75 measured temperature value.

Definition at line **156** of file **stm32072b_eval_tsensor.c**.

References **tsensor_drv**, and **TSENSORAddress**.

STM32072B_EVAL BSP User Manual

Private Variables

STM32072B_EVAL Common

Variables

GPIO_TypeDef *	**LED_PORT** [**LEDn**] LED variables.
const uint16_t	**LED_PIN** [**LEDn**]
GPIO_TypeDef *	**BUTTON_PORT** [**BUTTONn**] = {**TAMPER_BUTTON_GPIO_PORT**} BUTTON variables.
const uint16_t	**BUTTON_PIN** [**BUTTONn**] = {**TAMPER_BUTTON_PIN**}
const uint8_t	**BUTTON_IRQn** [**BUTTONn**] = {**TAMPER_BUTTON_EXTI_IRQn**}
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	**I2c1Timeout** = **EVAL_I2C1_TIMEOUT_MAX** BUS variables.
	I2c2Timeout =

uint32_t	**EVAL_I2C2_TIMEOUT_MAX**
I2C_HandleTypeDef	**heval_I2c1**
I2C_HandleTypeDef	**heval_I2c2**
uint32_t	**SpixTimeout** = **EVAL_SPIx_TIMEOUT_MAX**
static SPI_HandleTypeDef	**heval_Spi**

Variable Documentation

const uint8_t BUTTON_IRQn[BUTTONn] = {TAMPER_BUTTON_EXT

Definition at line 104 of file stm32072b_eval.c.

Referenced by BSP_PB_Init().

const uint16_t BUTTON_PIN[BUTTONn] = {TAMPER_BUTTON_PIN}

Definition at line 103 of file stm32072b_eval.c.

Referenced by BSP_PB_GetState(), and BSP_PB_Init().

GPIO_TypeDef* BUTTON_PORT[BUTTONn] = {TAMPER_BUTTON_(

BUTTON variables.

Definition at line 102 of file stm32072b_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 143 of file stm32072b_eval.c.

Referenced by BSP_COM_Init().

const uint16_t COM_RX_PIN[COMn] = {EVAL_COM1_RX_PIN}

Definition at line 139 of file stm32072b_eval.c.

Referenced by **BSP_COM_Init()**.

GPIO_TypeDef* COM_RX_PORT[COMn] = {EVAL_COM1_RX_GPIO_

Definition at line **135** of file **stm32072b_eval.c**.

Referenced by **BSP_COM_Init()**.

const uint16_t COM_TX_AF[COMn] = {EVAL_COM1_TX_AF}

Definition at line **141** of file **stm32072b_eval.c**.

Referenced by **BSP_COM_Init()**.

const uint16_t COM_TX_PIN[COMn] = {EVAL_COM1_TX_PIN}

Definition at line **137** of file **stm32072b_eval.c**.

Referenced by **BSP_COM_Init()**.

GPIO_TypeDef* COM_TX_PORT[COMn] = {EVAL_COM1_TX_GPIO_

Definition at line **133** of file **stm32072b_eval.c**.

Referenced by **BSP_COM_Init()**.

USART_TypeDef* COM_USART[COMn] = {EVAL_COM1}

COM variables.

Definition at line **131** of file **stm32072b_eval.c**.

Referenced by **BSP_COM_Init()**.

I2C_HandleTypeDef heval_I2c1

Definition at line **153** of file **stm32072b_eval.c**.

Referenced by **I2C1_Error()**, **I2C1_Init()**, **I2C1_IsDeviceReady()**, **I2C1_ReadBuffer()**, **I2C1_TransmitData()**, and **I2C1_WriteBuffer()**.

I2C_HandleTypeDef heval_I2c2

Definition at line **154** of file **stm32072b_eval.c**.

Referenced by **I2C2_Error()**, **I2C2_Init()**, and **I2C2_ReceiveData()**.

SPI_HandleTypeDef heval_Spi `[static]`

Definition at line **159** of file **stm32072b_eval.c**.

Referenced by **LCD_IO_WriteMultipleData()**, **SPIx_Error()**, **SPIx_FlushFifo()**, **SPIx_Init()**, **SPIx_Read()**, **SPIx_Write()**, and **SPIx_WriteReadData()**.

uint32_t I2c1Timeout = EVAL_I2C1_TIMEOUT_MAX

BUS variables.

Definition at line **151** of file **stm32072b_eval.c**.

Referenced by **I2C1_IsDeviceReady()**, **I2C1_ReadBuffer()**, **I2C1_TransmitData()**, and **I2C1_WriteBuffer()**.

uint32_t I2c2Timeout = EVAL_I2C2_TIMEOUT_MAX

Definition at line **152** of file **stm32072b_eval.c**.

Referenced by **I2C2_ReceiveData()**.

const uint8_t JOY_IRQn[JOYn]

Initial value:

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

Definition at line **121** of file **stm32072b_eval.c**.

Referenced by **BSP_JOY_Init()**.

const uint16_t JOY_PIN[JOYn]

Initial value:

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

Definition at line **115** of file **stm32072b_eval.c**.

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

GPIO_TypeDef* JOY_PORT[JOYn]

Initial value:

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

JOYSTICK variables.

Definition at line **109** of file **stm32072b_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 **95** of file **stm32072b_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 **90** of file **stm32072b_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 **158** of file **stm32072b_eval.c**.

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

STM32072B_EVAL BSP User Manual

Exported Types

STM32072B_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_TAMPER** = 0 } BUTTON Types Definition. More...
enum	**ButtonMode_TypeDef** { **BUTTON_MODE_GPIO** = 0, **BUTTON_MODE_EXTI** = 1 }

enum	**JOYState_TypeDef** {
	JOY_SEL = 0, **JOY_DOWN** = 1, **JOY_LEFT** = 2, **JOY_RIGHT** = 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_TAMPER

Definition at line **83** of file **stm32072b_eval.h**.

enum ButtonMode_TypeDef

Enumerator:

BUTTON_MODE_GPIO

BUTTON_MODE_EXTI

Definition at line **88** of file **stm32072b_eval.h**.

enum COM_TypeDef

COM Types Definition.

Enumerator:

COM1

Definition at line **116** of file **stm32072b_eval.h**.

enum JOYMode_TypeDef

Enumerator:

JOY_MODE_GPIO

JOY_MODE_EXTI

Definition at line **107** of file **stm32072b_eval.h**.

enum JOYState_TypeDef

JOYSTICK Types Definition.

Enumerator:

JOY_SEL
JOY_DOWN
JOY_LEFT
JOY_RIGHT
JOY_UP
JOY_NONE

Definition at line **97** of file **stm32072b_eval.h**.

enum Led_TypeDef

LED Types Definition.

Enumerator:

LED1
LED2
LED3
LED4
LED_GREEN
LED_ORANGE
LED_RED
LED_BLUE

Definition at line **67** of file **stm32072b_eval.h**.

Generated on Wed Jul 5 2017 08:56:11 for STM32072B_EVAL BSP
User Manual by doxygen 1.7.6.1

STM32072B_EVAL BSP User Manual

Defines

STM32072B_EVAL BUTTON

Exported Constants

Defines

#define	**JOYn** 5
#define	**BUTTONn** 1
#define	**TAMPER_BUTTON_PIN** GPIO_PIN_13 Tamper push-button.
#define	**TAMPER_BUTTON_GPIO_PORT** GPIOC
#define	**TAMPER_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_G
#define	**TAMPER_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_C
#define	**TAMPER_BUTTON_EXTI_IRQn** EXTI4_15_IRQn
#define	**TAMPERx_GPIO_CLK_ENABLE**(__BUTTON__) do { if((__B **BUTTON_TAMPER**) **TAMPER_BUTTON_GPIO_CLK_ENABL**
#define	**TAMPERx_GPIO_CLK_DISABLE**(__BUTTON__) (((__BUTT **BUTTON_TAMPER**) ? **TAMPER_BUTTON_GPIO_CLK_DISA**
#define	**RIGHT_JOY_PIN** GPIO_PIN_3 Joystick Right push-button.
#define	**RIGHT_JOY_GPIO_PORT** GPIOE
#define	**RIGHT_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_C
#define	**RIGHT_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_
#define	**RIGHT_JOY_EXTI_IRQn** EXTI2_3_IRQn
#define	**LEFT_JOY_PIN** GPIO_PIN_2 Joystick Left push-button.
#define	**LEFT_JOY_GPIO_PORT** GPIOE
#define	**LEFT_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_C
#define	**LEFT_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_C
#define	**LEFT_JOY_EXTI_IRQn** EXTI2_3_IRQn
#define	**UP_JOY_PIN** GPIO_PIN_9 Joystick Up push-button.
#define	**UP_JOY_GPIO_PORT** GPIOF
#define	**UP_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOF_CLK
#define	**UP_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOF_CLK
#define	**UP_JOY_EXTI_IRQn** EXTI4_15_IRQn
#define	**DOWN_JOY_PIN** GPIO_PIN_10

Joystick Down push-button.

#define **DOWN_JOY_GPIO_PORT** GPIOF

#define **DOWN_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOF_(

#define **DOWN_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOF_

#define **DOWN_JOY_EXTI_IRQn** EXTI4_15_IRQn

#define **SEL_JOY_PIN** GPIO_PIN_0

Joystick Sel push-button.

#define **SEL_JOY_GPIO_PORT** GPIOA

#define **SEL_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOA_CL

#define **SEL_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOA_CL

#define **SEL_JOY_EXTI_IRQn** EXTI0_1_IRQn

#define **JOYx_GPIO_CLK_ENABLE**(__JOY__)

#define **JOYx_GPIO_CLK_DISABLE**(__JOY__)

Define Documentation

#define BUTTONn 1

Definition at line 179 of file stm32072b_eval.h.

#define DOWN_JOY_EXTI_IRQn EXTI4_15_IRQn

Definition at line 228 of file stm32072b_eval.h.

#define DOWN_JOY_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOI

Definition at line 227 of file stm32072b_eval.h.

#define DOWN_JOY_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOF

Definition at line 226 of file stm32072b_eval.h.

#define DOWN_JOY_GPIO_PORT GPIOF

Definition at line 225 of file stm32072b_eval.h.

#define DOWN_JOY_PIN GPIO_PIN_10

Joystick Down push-button.

Definition at line 224 of file stm32072b_eval.h.

#define JOYn 5

Definition at line **178** of file **stm32072b_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_DOWN) ? DOWN_JOY_GPIO_CLK_DISABLE() :\
                                                  ((__JOY
__) == JOY_LEFT) ? LEFT_JOY_GPIO_CLK_DISABLE() :\
                                                  ((__JOY
__) == JOY_RIGHT) ? RIGHT_JOY_GPIO_CLK_DISABLE()
:\
                                                  ((__JOY
__) == JOY_UP) ? UP_JOY_GPIO_CLK_DISABLE() : 0 )
```

Definition at line **245** of file **stm32072b_eval.h**.

#define JOYx_GPIO_CLK_ENABLE (__JOY__)

Value:

```
do { if((__JOY__) == JOY_SEL) SEL_JOY_GPIO_CLK_EN
ABLE(); else \
                                                  if((__J
OY__) == JOY_DOWN) DOWN_JOY_GPIO_CLK_ENABLE(); el
se \
                                                  if((__J
OY__) == JOY_LEFT) LEFT_JOY_GPIO_CLK_ENABLE(); el
se \
                                                  if((__J
OY__) == JOY_RIGHT) RIGHT_JOY_GPIO_CLK_ENABLE();
else \
```

```
                                                    if((__J
OY__) == JOY_UP) UP_JOY_GPIO_CLK_ENABLE();} while
(0)
```

Definition at line **239** of file **stm32072b_eval.h**.

Referenced by **BSP_JOY_Init()**.

#define LEFT_JOY_EXTI_IRQn EXTI2_3_IRQn

Definition at line **210** of file **stm32072b_eval.h**.

#define LEFT_JOY_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOE_

Definition at line **209** of file **stm32072b_eval.h**.

#define LEFT_JOY_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOE_

Definition at line **208** of file **stm32072b_eval.h**.

#define LEFT_JOY_GPIO_PORT GPIOE

Definition at line **207** of file **stm32072b_eval.h**.

#define LEFT_JOY_PIN GPIO_PIN_2

Joystick Left push-button.

Definition at line **206** of file **stm32072b_eval.h**.

#define RIGHT_JOY_EXTI_IRQn EXTI2_3_IRQn

Definition at line 201 of file stm32072b_eval.h.

#define RIGHT_JOY_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOE

Definition at line 200 of file stm32072b_eval.h.

#define RIGHT_JOY_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOE

Definition at line 199 of file stm32072b_eval.h.

#define RIGHT_JOY_GPIO_PORT GPIOE

Definition at line 198 of file stm32072b_eval.h.

#define RIGHT_JOY_PIN GPIO_PIN_3

Joystick Right push-button.

Definition at line 197 of file stm32072b_eval.h.

#define SEL_JOY_EXTI_IRQn EXTI0_1_IRQn

Definition at line 237 of file stm32072b_eval.h.

#define SEL_JOY_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOA_

Definition at line 236 of file stm32072b_eval.h.

#define SEL_JOY_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOA_C

Definition at line **235** of file **stm32072b_eval.h**.

#define SEL_JOY_GPIO_PORT GPIOA

Definition at line **234** of file **stm32072b_eval.h**.

#define SEL_JOY_PIN GPIO_PIN_0

Joystick Sel push-button.

Definition at line **233** of file **stm32072b_eval.h**.

#define TAMPER_BUTTON_EXTI_IRQn EXTI4_15_IRQn

Definition at line **188** of file **stm32072b_eval.h**.

#define TAMPER_BUTTON_GPIO_CLK_DISABLE ( ) __HAL_RCC_

Definition at line **187** of file **stm32072b_eval.h**.

#define TAMPER_BUTTON_GPIO_CLK_ENABLE ( ) __HAL_RCC_

Definition at line **186** of file **stm32072b_eval.h**.

#define TAMPER_BUTTON_GPIO_PORT GPIOC

Definition at line **185** of file **stm32072b_eval.h**.

#define TAMPER_BUTTON_PIN GPIO_PIN_13

Tamper push-button.

Definition at line 184 of file stm32072b_eval.h.

#define TAMPERx_GPIO_CLK_DISABLE (__BUTTON__) (((__BU

Definition at line 192 of file stm32072b_eval.h.

#define TAMPERx_GPIO_CLK_ENABLE (__BUTTON__) do { if((

Definition at line 190 of file stm32072b_eval.h.

Referenced by BSP_PB_Init().

#define UP_JOY_EXTI_IRQn EXTI4_15_IRQn

Definition at line 219 of file stm32072b_eval.h.

#define UP_JOY_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOF_C

Definition at line 218 of file stm32072b_eval.h.

#define UP_JOY_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOF_CL

Definition at line 217 of file stm32072b_eval.h.

#define UP_JOY_GPIO_PORT GPIOF

Definition at line 216 of file stm32072b_eval.h.

#define UP_JOY_PIN GPIO_PIN_9

Joystick Up push-button.

Definition at line **215** of file **stm32072b_eval.h**.

STM32072B_EVAL BSP User Manual

Defines

STM32072B_EVAL COM

Exported Constants

Defines

#define	**COMn** 1
#define	**EVAL_COM1** USART2 Definition for COM port1, connected to USART2.
#define	**EVAL_COM1_CLK_ENABLE**() __HAL_RCC_USART2_CLK_
#define	**EVAL_COM1_CLK_DISABLE**() __HAL_RCC_USART2_CLK
#define	**EVAL_COM1_TX_PIN** GPIO_PIN_5
#define	**EVAL_COM1_TX_GPIO_PORT** GPIOD
#define	**EVAL_COM1_TX_GPIO_CLK_ENABLE**() __HAL_RCC_GPI
#define	**EVAL_COM1_TX_GPIO_CLK_DISABLE**() __HAL_RCC_GP
#define	**EVAL_COM1_TX_AF** GPIO_AF0_USART2
#define	**EVAL_COM1_RX_PIN** GPIO_PIN_6
#define	**EVAL_COM1_RX_GPIO_PORT** GPIOD
#define	**EVAL_COM1_RX_GPIO_CLK_ENABLE**() __HAL_RCC_GPI
#define	**EVAL_COM1_RX_GPIO_CLK_DISABLE**() __HAL_RCC_GP
#define	**EVAL_COM1_RX_AF** GPIO_AF0_USART2
#define	**EVAL_COM1_CTS_PIN** GPIO_PIN_3
#define	**EVAL_COM1_CTS_GPIO_PORT** GPIOD
#define	**EVAL_COM1_CTS_GPIO_CLK_ENABLE**() __HAL_RCC_GI
#define	**EVAL_COM1_CTS_GPIO_CLK_DISABLE**() __HAL_RCC_G
#define	**EVAL_COM1_CTS_AF** GPIO_AF0_USART2
#define	**EVAL_COM1_RTS_PIN** GPIO_PIN_4
#define	**EVAL_COM1_RTS_GPIO_PORT** GPIOD
#define	**EVAL_COM1_RTS_GPIO_CLK_ENABLE**() __HAL_RCC_GI
#define	**EVAL_COM1_RTS_GPIO_CLK_DISABLE**() __HAL_RCC_G
#define	**EVAL_COM1_RTS_AF** GPIO_AF0_USART2
#define	**EVAL_COM1_IRQn** USART2_IRQn
#define	**COMx_CLK_ENABLE**(__COM__) do { if((__COM__) == **COM** **EVAL_COM1_CLK_ENABLE**();} while(0)
#define	**COMx_CLK_DISABLE**(__COM__) (((__COM__) == **COM1**) **EVAL_COM1_CLK_DISABLE**() : 0)
#define	**COMx_TX_GPIO_CLK_ENABLE**(__COM__) do { if((__COM

	EVAL_COM1_TX_GPIO_CLK_ENABLE();} while(0)
#define	**COMx_TX_GPIO_CLK_DISABLE**(__COM__) (((__COM__) = **EVAL_COM1_TX_GPIO_CLK_DISABLE**() : 0)
#define	**COMx_RX_GPIO_CLK_ENABLE**(__COM__) do { if((__COM **EVAL_COM1_RX_GPIO_CLK_ENABLE**();} while(0)
#define	**COMx_RX_GPIO_CLK_DISABLE**(__COM__) (((__COM__) **EVAL_COM1_RX_GPIO_CLK_DISABLE**() : 0)
#define	**COMx_CTS_GPIO_CLK_ENABLE**(__COM__) do { if((__CO **EVAL_COM1_CTS_GPIO_CLK_ENABLE**();} while(0)
#define	**COMx_CTS_GPIO_CLK_DISABLE**(__COM__) (((__COM__ **EVAL_COM1_CTS_GPIO_CLK_DISABLE**() : 0)
#define	**COMx_RTS_GPIO_CLK_ENABLE**(__COM__) do { if((__CO **EVAL_COM1_RTS_GPIO_CLK_ENABLE**();} while(0)
#define	**COMx_RTS_GPIO_CLK_DISABLE**(__COM__) (((__COM__ **EVAL_COM1_RTS_GPIO_CLK_DISABLE**() : 0)

Define Documentation

#define COMn 1

Definition at line 258 of file stm32072b_eval.h.

#define COMx_CLK_DISABLE (__COM__) (((__COM__) == COM

Definition at line 294 of file stm32072b_eval.h.

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

Definition at line 293 of file stm32072b_eval.h.

Referenced by BSP_COM_Init().

#define COMx_CTS_GPIO_CLK_DISABLE (__COM__) (((__COM

Definition at line 303 of file stm32072b_eval.h.

#define COMx_CTS_GPIO_CLK_ENABLE (__COM__) do { if((__

Definition at line 302 of file stm32072b_eval.h.

#define COMx_RTS_GPIO_CLK_DISABLE (__COM__) (((__COM

Definition at line 306 of file stm32072b_eval.h.

#define COMx_RTS_GPIO_CLK_ENABLE (__COM__) do { if((__

Definition at line 305 of file stm32072b_eval.h.

#define COMx_RX_GPIO_CLK_DISABLE (__COM__) (((__COM_

Definition at line 300 of file stm32072b_eval.h.

#define COMx_RX_GPIO_CLK_ENABLE (__COM__) do { if((__C

Definition at line 299 of file stm32072b_eval.h.

Referenced by BSP_COM_Init().

#define COMx_TX_GPIO_CLK_DISABLE (__COM__) (((__COM_

Definition at line 297 of file stm32072b_eval.h.

#define COMx_TX_GPIO_CLK_ENABLE (__COM__) do { if((__C

Definition at line 296 of file stm32072b_eval.h.

Referenced by BSP_COM_Init().

#define EVAL_COM1 USART2

Definition for COM port1, connected to USART2.

Definition at line 263 of file stm32072b_eval.h.

#define EVAL_COM1_CLK_DISABLE ( ) __HAL_RCC_USART2_CL

Definition at line 265 of file stm32072b_eval.h.

#define EVAL_COM1_CLK_ENABLE ( ) __HAL_RCC_USART2_CL

Definition at line 264 of file stm32072b_eval.h.

#define EVAL_COM1_CTS_AF GPIO_AF0_USART2

Definition at line 283 of file stm32072b_eval.h.

#define EVAL_COM1_CTS_GPIO_CLK_DISABLE ( ) __HAL_RCC_

Definition at line 282 of file stm32072b_eval.h.

#define EVAL_COM1_CTS_GPIO_CLK_ENABLE ( ) __HAL_RCC_

Definition at line 281 of file stm32072b_eval.h.

#define EVAL_COM1_CTS_GPIO_PORT GPIOD

Definition at line 280 of file stm32072b_eval.h.

#define EVAL_COM1_CTS_PIN GPIO_PIN_3

Definition at line 279 of file stm32072b_eval.h.

#define EVAL_COM1_IRQn USART2_IRQn

Definition at line 291 of file stm32072b_eval.h.

#define EVAL_COM1_RTS_AF GPIO_AF0_USART2

Definition at line 289 of file stm32072b_eval.h.

#define EVAL_COM1_RTS_GPIO_CLK_DISABLE ( ) __HAL_RCC_

Definition at line 288 of file stm32072b_eval.h.

#define EVAL_COM1_RTS_GPIO_CLK_ENABLE ( ) __HAL_RCC_(

Definition at line 287 of file stm32072b_eval.h.

#define EVAL_COM1_RTS_GPIO_PORT GPIOD

Definition at line 286 of file stm32072b_eval.h.

#define EVAL_COM1_RTS_PIN GPIO_PIN_4

Definition at line 285 of file stm32072b_eval.h.

#define EVAL_COM1_RX_AF GPIO_AF0_USART2

Definition at line 277 of file stm32072b_eval.h.

#define EVAL_COM1_RX_GPIO_CLK_DISABLE ( ) __HAL_RCC_G

Definition at line 276 of file stm32072b_eval.h.

#define EVAL_COM1_RX_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line **275** of file **stm32072b_eval.h**.

#define EVAL_COM1_RX_GPIO_PORT GPIOD

Definition at line **274** of file **stm32072b_eval.h**.

#define EVAL_COM1_RX_PIN GPIO_PIN_6

Definition at line **273** of file **stm32072b_eval.h**.

#define EVAL_COM1_TX_AF GPIO_AF0_USART2

Definition at line **271** of file **stm32072b_eval.h**.

#define EVAL_COM1_TX_GPIO_CLK_DISABLE ( ) __HAL_RCC_G

Definition at line **270** of file **stm32072b_eval.h**.

#define EVAL_COM1_TX_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line **269** of file **stm32072b_eval.h**.

#define EVAL_COM1_TX_GPIO_PORT GPIOD

Definition at line **268** of file **stm32072b_eval.h**.

#define EVAL_COM1_TX_PIN GPIO_PIN_5

Definition at line **267** of file **stm32072b_eval.h**.

Generated on Wed Jul 5 2017 08:56:11 for STM32072B_EVAL BSP
User Manual by doxygen 1.7.6.1

STM32072B_EVAL BSP User Manual

Private Types

STM32072B_EVAL EEPROM

Variables

EEPROM_DrvTypeDef	EEPROM_I2C_Drv

Variable Documentation

EEPROM_DrvTypeDef EEPROM_I2C_Drv

Initial value:

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

Definition at line **122** of file **stm32072b_eval_eeprom.c**.

Referenced by **BSP_EEPROM_Init()**.

STM32072B_EVAL BSP User Manual

Private Functions

STM32072B_EVAL EEPROM

Functions

static uint32_t	**EEPROM_I2C_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
static uint32_t	**EEPROM_I2C_ReadBuffer** (uint8_t *pBuffer, uint16_t ReadAddr, uint32_t *NumByteToRead) Reads a block of data from the I2C EEPROM.
static uint32_t	**EEPROM_I2C_WritePage** (uint8_t *pBuffer, uint16_t WriteAddr, uint32_t *NumByteToWrite) Writes more than one byte to the EEPROM with a single WRITE cycle.
static uint32_t	**EEPROM_I2C_WaitEepromStandbyState** (void) Wait for EEPROM I2C Standby state.

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 **367** of file **stm32072b_eval_eeprom.c**.

References **EEPROM_ADDRESS_M24LR64_A01**, **EEPROM_ADDRESS_M24LR64_A02**, **EEPROM_FAIL**, **EEPROM_IO_Init()**, **EEPROM_IO_IsDeviceReady()**, **EEPROM_MAX_TRIALS**, **EEPROM_OK**, **EEPROM_PAGESIZE_M24LR64**, **EEPROMAddress**, and **EEPROMPageSize**.

static uint32_t EEPROM_I2C_ReadBuffer (uint8_t * pBuffer, uint16_t ReadAddr, uint32_t * NumByteToR[illegible]) [static]

Reads a block of data from the I2C EEPROM.

Parameters:

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

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

Return values:

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

Definition at line **399** of file **stm32072b_eval_eeprom.c**.

References **EEPROM_FAIL**, **EEPROM_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 **467** of file **stm32072b_eval_eeprom.c**.

References **BSP_EEPROM_TIMEOUT_UserCallback()**,

EEPROM_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 **436** of file **stm32072b_eval_eeprom.c**.

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

STM32072B_EVAL BSP User Manual

Functions

LINK Operations Functions

STM32072B_EVAL Common

Functions

void	**EEPROM_IO_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_IO_WriteData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Write data to I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_IO_ReadData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Read data from I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_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	**HDMI_CEC_IO_Init** (void) Initializes CEC low level.

HAL_StatusTypeDef	**HDMI_CEC_IO_WriteData** (uint8_t *pBuffer, uint16_t BufferSize) Write data to I2C HDMI CEC driver.
HAL_StatusTypeDef	**HDMI_CEC_IO_ReadData** (uint16_t DevAddress, uint8_t *pBuffer, uint16_t BufferSize) Read data to I2C HDMI CEC driver.
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) Writes address on LCD register.
uint16_t	**LCD_IO_ReadData** (uint16_t Reg) Read data from LCD data register.
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_CSState** (uint8_t state)
void	**SD_IO_WriteReadData** (const uint8_t *DataIn, uint8_t *DataOut, uint16_t DataLength) Write a byte on the SD.
uint8_t	**SD_IO_WriteByte** (uint8_t Data) Writes a byte on the SD.

Function Documentation

void EEPROM_IO_Init (void)

Initializes peripherals used by the I2C EEPROM driver.

Return values:
None

Definition at line 1207 of file stm32072b_eval.c.

References I2C1_Init().

Referenced by EEPROM_I2C_Init().

HAL_StatusTypeDef EEPROM_IO_IsDeviceReady (uint16_t DevAd
uint32_t Trials
)

Checks if target device is ready for communication.

Note:
This function is used with Memory devices

Parameters:

DevAddress	Target device address
Trials	Number of trials

Return values:
HAL status

Definition at line 1245 of file stm32072b_eval.c.

References I2C1_IsDeviceReady().

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

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

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 **1233** of file **stm32072b_eval.c**.

References **I2C1_ReadBuffer()**.

Referenced by **EEPROM_I2C_ReadBuffer()**.

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

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 **1220** of file **stm32072b_eval.c**.

References **I2C1_WriteBuffer()**.

Referenced by **EEPROM_I2C_WritePage()**.

void HDMI_CEC_IO_Init (void)

Initializes CEC low level.

Return values:

None

Definition at line **1302** of file **stm32072b_eval.c**.

References **HDMI_CEC_HPD_SINK_CLK_ENABLE**, **HDMI_CEC_HPD_SINK_GPIO_PORT**, **HDMI_CEC_HPD_SINK_PIN**, **HDMI_CEC_HPD_SOURCE_CLK_ENABLE**, **HDMI_CEC_HPD_SOURCE_GPIO_PORT**, **HDMI_CEC_HPD_SOURCE_PIN**, **HDMI_CEC_IRQn**, **HDMI_CEC_LINE_AF**, **HDMI_CEC_LINE_CLK_ENABLE**, **HDMI_CEC_LINE_GPIO_PORT**, **HDMI_CEC_LINE_PIN**, **I2C1_Init()**, and **I2C2_Init()**.

HAL_StatusTypeDef HDMI_CEC_IO_ReadData (uint16_t DevAddre
uint8_t * pBuffer,

uint16_t BufferSize
)

Read data to I2C HDMI CEC driver.

Parameters:

DevAddress	Target device address
pBuffer	Pointer to data buffer
BufferSize	Amount of data to be sent

Return values:

HAL	status

Definition at line **1367** of file **stm32072b_eval.c**.

References **I2C2_ReceiveData()**.

HAL_StatusTypeDef **HDMI_CEC_IO_WriteData** (**uint8_t** * **pBuffer,**
uint16_t **BufferSize**
)

Write data to I2C HDMI CEC driver.

Parameters:

pBuffer	Pointer to data buffer
BufferSize	Amount of data to be sent

Return values:

HAL	status

Definition at line **1355** of file **stm32072b_eval.c**.

References **I2C1_TransmitData()**.

void LCD_Delay (uint32_t Delay)

Wait for loop in ms.

Parameters:

Delay in ms.

Definition at line **1097** of file **stm32072b_eval.c**.

void LCD_IO_Init (void)

Configures the LCD_SPI interface.

Return values:

None

Definition at line **967** of file **stm32072b_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 data from LCD data register.

Parameters:

Reg Regsiter to be read

Return values:

readvalue

Definition at line **1067** of file **stm32072b_eval.c**.

References **LCD_CS_HIGH**, **LCD_CS_LOW**, **LCD_IO_WriteReg()**,

LCD_READ_REG, SPIx_Read(), SPIx_Write(), and START_BYTE.

void LCD_IO_WriteMultipleData (uint8_t * pData,
uint32_t Size
)

Write register value.

Parameters:

pData Pointer on the register value

Size Size of byte to transmit to the register

Return values:

None

Definition at line 994 of file stm32072b_eval.c.

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

void LCD_IO_WriteReg (uint8_t Reg)

Writes address on LCD register.

Parameters:

Reg Register to be written

Return values:

None

Definition at line 1046 of file stm32072b_eval.c.

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

Referenced by LCD_IO_ReadData().

void SD_IO_CSState (uint8_t state)

Definition at line 1160 of file stm32072b_eval.c.

References SD_CS_HIGH, and SD_CS_LOW.

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

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 1109 of file stm32072b_eval.c.

References LCD_CS_HIGH, LCD_NCS_GPIO_PORT, LCD_NCS_PIN, 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_WriteByte (uint8_t Data)

Writes a byte on the SD.

Parameters:

Data byte to send.

Return values:

None

Definition at line **1190** of file **stm32072b_eval.c**.

References **SPIx_WriteReadData()**.

Referenced by **BSP_SD_Erase()**, **BSP_SD_GetStatus()**, **BSP_SD_ReadBlocks()**, **BSP_SD_WriteBlocks()**, **SD_GetCIDRegister()**, **SD_GetCSDRegister()**, **SD_GetDataResponse()**, **SD_GoIdleState()**, **SD_IO_Init()**, **SD_ReadData()**, **SD_SendCmd()**, and **SD_WaitData()**.

---

void SD_IO_WriteReadData (const uint8_t * DataIn,
uint8_t * DataOut,
uint16_t DataLength
)

Write a byte on the SD.

Parameters:

DataIn byte to send.

DataOut read byte.

DataLength data length.

Return values:

None

Definition at line **1179** of file **stm32072b_eval.c**.

References **SPIx_WriteReadData()**.

Referenced by **BSP_SD_ReadBlocks()**, **BSP_SD_WriteBlocks()**, and **SD_SendCmd()**.

void TSENSOR_IO_Init (void)

Initializes peripherals used by the I2C Temperature Sensor driver.

Return values:

None

Definition at line **1255** of file **stm32072b_eval.c**.

References **I2C1_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 **1292** of file **stm32072b_eval.c**.

References **I2C1_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 **1281** of file **stm32072b_eval.c**.

References **I2C1_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 **1268** of file **stm32072b_eval.c**.

References **I2C1_WriteBuffer()**.

STM32072B_EVAL BSP User Manual

Functions

LINK Operations Functions

STM32072B_EVAL EEPROM

Functions

void	**EEPROM_IO_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_IO_WriteData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Write data to I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_IO_ReadData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Read data from I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_IO_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.

Function Documentation

void EEPROM_IO_Init (void)

Initializes peripherals used by the I2C EEPROM driver.

Return values:
None

Definition at line 1207 of file stm32072b_eval.c.

References I2C1_Init().

Referenced by EEPROM_I2C_Init().

HAL_StatusTypeDef EEPROM_IO_IsDeviceReady (uint16_t DevAd
uint32_t Trials
)

Checks if target device is ready for communication.

Note:
This function is used with Memory devices

Parameters:

DevAddress	Target device address
Trials	Number of trials

Return values:
HAL status

Definition at line 1245 of file stm32072b_eval.c.

References I2C1_IsDeviceReady().

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

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

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 **1233** of file **stm32072b_eval.c**.

References **I2C1_ReadBuffer()**.

Referenced by **EEPROM_I2C_ReadBuffer()**.

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

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 **1220** of file **stm32072b_eval.c**.

References **I2C1_WriteBuffer()**.

Referenced by **EEPROM_I2C_WritePage()**.

STM32072B_EVAL BSP User Manual

Private Variables

STM32072B_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 101 of file stm32072b_eval_eeprom.c.

__IO uint16_t EEPROMAddress = 0

Definition at line 96 of file stm32072b_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 98 of file stm32072b_eval_eeprom.c.

__IO uint8_t EEPROMDataWrite

Definition at line 99 of file stm32072b_eval_eeprom.c.

__IO uint16_t EEPROMPageSize = 0

Definition at line 97 of file stm32072b_eval_eeprom.c.

Referenced by BSP_EEPROM_WriteBuffer(), and EEPROM_I2C_Init().

User Manual by  1.7.6.1

STM32072B_EVAL BSP User Manual

Defines

STM32072B_EVAL BUS

Exported Constants

Defines

#define **EVAL_I2C1** I2C1
#define **EVAL_I2C1_CLK_ENABLE**() __HAL_RCC_I2C1_CLK_ENA
#define **EVAL_I2C1_CLK_DISABLE**() __HAL_RCC_I2C1_CLK_DIS
#define **EVAL_I2C1_FORCE_RESET**() __HAL_RCC_I2C1_FORCE_
#define **EVAL_I2C1_RELEASE_RESET**() __HAL_RCC_I2C1_RELE
#define **EVAL_I2C1_SCL_PIN** GPIO_PIN_6 /* PB.6 */
#define **EVAL_I2C1_SDA_PIN** GPIO_PIN_7 /* PB.7 */
#define **EVAL_I2C1_GPIO_PORT** GPIOB /* GPIOB */
#define **EVAL_I2C1_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_C
#define **EVAL_I2C1_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_
#define **EVAL_I2C1_SCL_SDA_AF** GPIO_AF1_I2C1
#define **EVAL_I2C2** I2C2
#define **EVAL_I2C2_CLK_ENABLE**() __HAL_RCC_I2C2_CLK_ENA
#define **EVAL_I2C2_CLK_DISABLE**() __HAL_RCC_I2C2_CLK_DIS
#define **EVAL_I2C2_FORCE_RESET**() __HAL_RCC_I2C2_FORCE_
#define **EVAL_I2C2_RELEASE_RESET**() __HAL_RCC_I2C2_RELE
#define **EVAL_I2C2_SCL_PIN** GPIO_PIN_13 /* PB.13 */
#define **EVAL_I2C2_SDA_PIN** GPIO_PIN_14 /* PB.14 */
#define **EVAL_I2C2_GPIO_PORT** GPIOB /* GPIOB */
#define **EVAL_I2C2_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_C
#define **EVAL_I2C2_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_
#define **EVAL_I2C2_AF** GPIO_AF5_I2C2
#define **EVAL_I2C2_IRQn** I2C2_IRQn
#define **EVAL_I2C1_TIMEOUT_MAX** 1000
#define **EVAL_I2C2_TIMEOUT_MAX** 1000
#define **I2C2_TIMING** 0x00E0D3FF
#define **I2C1_TIMING** 0x00E0D3FF
#define **EVAL_SPIx** SPI1
Definition for SPI Interface pins (SPI1 used)
#define **EVAL_SPIx_CLK_ENABLE**() __HAL_RCC_SPI1_CLK_ENA
#define **EVAL_SPIx_CLK_DISABLE**() __HAL_RCC_SPI1_CLK_DIS

```
#define EVAL_SPIx_FORCE_RESET()   __HAL_RCC_SPI1_FORCE_
#define EVAL_SPIx_RELEASE_RESET()   __HAL_RCC_SPI1_RELE
#define EVAL_SPIx_SCK_PIN   GPIO_PIN_3 /* PB.03 */
#define EVAL_SPIx_SCK_GPIO_PORT   GPIOB /* GPIOB */
#define EVAL_SPIx_SCK_GPIO_CLK_ENABLE()   __HAL_RCC_GPI
#define EVAL_SPIx_SCK_GPIO_CLK_DISABLE()   __HAL_RCC_GP
#define EVAL_SPIx_SCK_AF   GPIO_AF0_SPI1
#define EVAL_SPIx_MISO_PIN   GPIO_PIN_14 /* PE.14 */
#define EVAL_SPIx_MISO_GPIO_PORT   GPIOE /* GPIOE */
#define EVAL_SPIx_MISO_GPIO_CLK_ENABLE()   __HAL_RCC_GP
#define EVAL_SPIx_MISO_GPIO_CLK_DISABLE()   __HAL_RCC_G
#define EVAL_SPIx_MISO_AF   GPIO_AF1_SPI1
#define EVAL_SPIx_MOSI_PIN   GPIO_PIN_15 /* PE.15 */
#define EVAL_SPIx_MOSI_GPIO_PORT   GPIOE /* GPIOE */
#define EVAL_SPIx_MOSI_GPIO_CLK_ENABLE()   __HAL_RCC_GP
#define EVAL_SPIx_MOSI_GPIO_CLK_DISABLE()   __HAL_RCC_G
#define EVAL_SPIx_MOSI_AF   GPIO_AF1_SPI1
#define EVAL_SPIx_MOSI_DIR_PIN   GPIO_PIN_2 /* PB.02 */
#define EVAL_SPIx_MOSI_DIR_GPIO_PORT   GPIOB /* GPIOB */
#define EVAL_SPIx_MOSI_DIR_GPIO_CLK_ENABLE()   __HAL_RC
#define EVAL_SPIx_MOSI_DIR_GPIO_CLK_DISABLE()   __HAL_RC
#define EVAL_SPIx_TIMEOUT_MAX   1000
```

Define Documentation

#define EVAL_I2C1 I2C1

Definition at line 318 of file stm32072b_eval.h.

Referenced by I2C1_Init().

#define EVAL_I2C1_CLK_DISABLE ( ) __HAL_RCC_I2C1_CLK_DI

Definition at line 320 of file stm32072b_eval.h.

#define EVAL_I2C1_CLK_ENABLE ( ) __HAL_RCC_I2C1_CLK_EN

Definition at line 319 of file stm32072b_eval.h.

Referenced by I2C1_MspInit().

#define EVAL_I2C1_FORCE_RESET ( ) __HAL_RCC_I2C1_FORCE

Definition at line 321 of file stm32072b_eval.h.

Referenced by I2C1_MspInit().

#define EVAL_I2C1_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOB

Definition at line 329 of file stm32072b_eval.h.

#define EVAL_I2C1_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOB_

Definition at line 328 of file stm32072b_eval.h.

Referenced by **I2C1_MspInit()**.

#define EVAL_I2C1_GPIO_PORT GPIOB /* GPIOB */

Definition at line **327** of file **stm32072b_eval.h**.

Referenced by **I2C1_MspInit()**.

#define EVAL_I2C1_RELEASE_RESET ( ) __HAL_RCC_I2C1_REL

Definition at line **322** of file **stm32072b_eval.h**.

Referenced by **I2C1_MspInit()**.

#define EVAL_I2C1_SCL_PIN GPIO_PIN_6 /* PB.6 */

Definition at line **324** of file **stm32072b_eval.h**.

Referenced by **I2C1_MspInit()**.

#define EVAL_I2C1_SCL_SDA_AF GPIO_AF1_I2C1

Definition at line **330** of file **stm32072b_eval.h**.

Referenced by **I2C1_MspInit()**.

#define EVAL_I2C1_SDA_PIN GPIO_PIN_7 /* PB.7 */

Definition at line **325** of file **stm32072b_eval.h**.

Referenced by **I2C1_MspInit()**.

#define EVAL_I2C1_TIMEOUT_MAX 1000

Definition at line 355 of file stm32072b_eval.h.

#define EVAL_I2C2 I2C2

Definition at line 333 of file stm32072b_eval.h.

Referenced by I2C2_Init().

#define EVAL_I2C2_AF GPIO_AF5_I2C2

Definition at line 345 of file stm32072b_eval.h.

Referenced by I2C2_MspInit().

#define EVAL_I2C2_CLK_DISABLE ( ) __HAL_RCC_I2C2_CLK_DI

Definition at line 335 of file stm32072b_eval.h.

#define EVAL_I2C2_CLK_ENABLE ( ) __HAL_RCC_I2C2_CLK_EN

Definition at line 334 of file stm32072b_eval.h.

Referenced by I2C2_MspInit().

#define EVAL_I2C2_FORCE_RESET ( ) __HAL_RCC_I2C2_FORCE

Definition at line 336 of file stm32072b_eval.h.

Referenced by I2C2_MspInit().

#define EVAL_I2C2_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOB

Definition at line 344 of file stm32072b_eval.h.

#define EVAL_I2C2_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOB_

Definition at line 343 of file stm32072b_eval.h.

Referenced by I2C2_MspInit().

#define EVAL_I2C2_GPIO_PORT GPIOB /* GPIOB */

Definition at line 342 of file stm32072b_eval.h.

Referenced by I2C2_MspInit().

#define EVAL_I2C2_IRQn I2C2_IRQn

Definition at line 348 of file stm32072b_eval.h.

#define EVAL_I2C2_RELEASE_RESET ( ) __HAL_RCC_I2C2_REL

Definition at line 337 of file stm32072b_eval.h.

Referenced by I2C2_MspInit().

#define EVAL_I2C2_SCL_PIN GPIO_PIN_13 /* PB.13 */

Definition at line 339 of file stm32072b_eval.h.

Referenced by I2C2_MspInit().

#define EVAL_I2C2_SDA_PIN GPIO_PIN_14 /* PB.14 */

Definition at line 340 of file stm32072b_eval.h.

Referenced by I2C2_MspInit().

#define EVAL_I2C2_TIMEOUT_MAX 1000

Definition at line 356 of file stm32072b_eval.h.

#define EVAL_SPIx SPI1

Definition for SPI Interface pins (SPI1 used)

Definition at line 369 of file stm32072b_eval.h.

Referenced by SPIx_Init().

#define EVAL_SPIx_CLK_DISABLE ( ) __HAL_RCC_SPI1_CLK_DI

Definition at line 371 of file stm32072b_eval.h.

#define EVAL_SPIx_CLK_ENABLE ( ) __HAL_RCC_SPI1_CLK_EN

Definition at line 370 of file stm32072b_eval.h.

Referenced by SPIx_MspInit().

#define EVAL_SPIx_FORCE_RESET ( ) __HAL_RCC_SPI1_FORCI

Definition at line 372 of file stm32072b_eval.h.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_MISO_AF GPIO_AF1_SPI1

Definition at line **385** of file **stm32072b_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_MISO_GPIO_CLK_DISABLE ( ) __HAL_RCC_

Definition at line **384** of file **stm32072b_eval.h**.

#define EVAL_SPIx_MISO_GPIO_CLK_ENABLE ( ) __HAL_RCC_(

Definition at line **383** of file **stm32072b_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_MISO_GPIO_PORT GPIOE /* GPIOE */

Definition at line **382** of file **stm32072b_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_MISO_PIN GPIO_PIN_14 /* PE.14 */

Definition at line **381** of file **stm32072b_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_MOSI_AF GPIO_AF1_SPI1

Definition at line 391 of file stm32072b_eval.h.

Referenced by SPIx_MspInit().

#define EVAL_SPIx_MOSI_DIR_GPIO_CLK_DISABLE ( ) __HAL_R

Definition at line 396 of file stm32072b_eval.h.

#define EVAL_SPIx_MOSI_DIR_GPIO_CLK_ENABLE ( ) __HAL_R

Definition at line 395 of file stm32072b_eval.h.

Referenced by SPIx_MspInit().

#define EVAL_SPIx_MOSI_DIR_GPIO_PORT GPIOB /* GPIOB */

Definition at line 394 of file stm32072b_eval.h.

Referenced by SPIx_MspInit().

#define EVAL_SPIx_MOSI_DIR_PIN GPIO_PIN_2 /* PB.02 */

Definition at line 393 of file stm32072b_eval.h.

Referenced by SPIx_MspInit().

#define EVAL_SPIx_MOSI_GPIO_CLK_DISABLE ( ) __HAL_RCC_

Definition at line 390 of file stm32072b_eval.h.

#define EVAL_SPIx_MOSI_GPIO_CLK_ENABLE ( ) __HAL_RCC_

Definition at line **389** of file **stm32072b_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_MOSI_GPIO_PORT GPIOE /* GPIOE */

Definition at line **388** of file **stm32072b_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_MOSI_PIN GPIO_PIN_15 /* PE.15 */

Definition at line **387** of file **stm32072b_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_RELEASE_RESET ( ) __HAL_RCC_SPI1_REL

Definition at line **373** of file **stm32072b_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_SCK_AF GPIO_AF0_SPI1

Definition at line **379** of file **stm32072b_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_SCK_GPIO_CLK_DISABLE ( ) __HAL_RCC_C

Definition at line **378** of file **stm32072b_eval.h**.

#define EVAL_SPIx_SCK_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line 377 of file stm32072b_eval.h.

Referenced by SPIx_MspInit().

#define EVAL_SPIx_SCK_GPIO_PORT GPIOB /* GPIOB */

Definition at line 376 of file stm32072b_eval.h.

Referenced by SPIx_MspInit().

#define EVAL_SPIx_SCK_PIN GPIO_PIN_3 /* PB.03 */

Definition at line 375 of file stm32072b_eval.h.

Referenced by SPIx_MspInit().

#define EVAL_SPIx_TIMEOUT_MAX 1000

Definition at line 403 of file stm32072b_eval.h.

#define I2C1_TIMING 0x00E0D3FF

Definition at line 361 of file stm32072b_eval.h.

Referenced by I2C1_Init().

#define I2C2_TIMING 0x00E0D3FF

Definition at line 360 of file stm32072b_eval.h.

Referenced by **I2C2_Init()**.

STM32072B_EVAL BSP User Manual

Private Variables

STM32072B_EVAL SD

Variables

__IO uint8_t	**SdStatus** = **SD_NOT_PRESENT**
uint16_t	**flag_SDHC** = 0

Variable Documentation

uint16_t flag_SDHC = 0

Definition at line 244 of file stm32072b_eval_sd.c.

__IO uint8_t SdStatus = SD_NOT_PRESENT

Definition at line 238 of file stm32072b_eval_sd.c.

STM32072B_EVAL BSP User Manual

Defines

STM32072B_EVAL COMPONENT

Exported Constants

Defines

#define	**LCD_CS_LOW**() HAL_GPIO_WritePin(**LCD_NCS_GPIO_PO** GPIO_PIN_RESET)
#define	**LCD_CS_HIGH**() HAL_GPIO_WritePin(**LCD_NCS_GPIO_PC** GPIO_PIN_SET)
#define	**LCD_NCS_PIN** GPIO_PIN_6 /* PE. 06*/ LCD Control pins.
#define	**LCD_NCS_GPIO_PORT** GPIOE /* 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_2 /* PF.2 */ SD card Control pin.
#define	**SD_CS_GPIO_PORT** GPIOF /* GPIOF */
#define	**SD_CS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOF_CLK_
#define	**SD_CS_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOF_CLK_
#define	**SD_DETECT_PIN** GPIO_PIN_15 /* PB.15 */ SD Detect Interface pins.
#define	**SD_DETECT_GPIO_PORT** GPIOB /* GPIOB */
#define	**SD_DETECT_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_
#define	**SD_DETECT_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_
#define	**SD_DETECT_EXTI_IRQn** EXTI4_15_IRQn
#define	**HDMI_CEC_HPD_SINK_PIN** GPIO_PIN_15 /* PD.15 */ I2C HDMI CEC Interface pins.
#define	**HDMI_CEC_HPD_SINK_GPIO_PORT** GPIOD
#define	**HDMI_CEC_HPD_SINK_CLK_ENABLE**() __HAL_RCC_GPI
#define	**HDMI_CEC_HPD_SINK_CLK_DISABLE**() __HAL_RCC_GP
#define	**HDMI_CEC_HPD_SOURCE_PIN** GPIO_PIN_0 /* PE.0 */
#define	**HDMI_CEC_HPD_SOURCE_GPIO_PORT** GPIOE

```
#define HDMI_CEC_HPD_SOURCE_CLK_ENABLE()  __HAL_RCC_
#define HDMI_CEC_HPD_SOURCE_CLK_DISABLE()  __HAL_RCC_
#define HDMI_CEC_LINE_PIN  GPIO_PIN_8 /* PB.8 */
#define HDMI_CEC_LINE_GPIO_PORT  GPIOB
#define HDMI_CEC_LINE_CLK_ENABLE()  __HAL_RCC_GPIOB_CL
#define HDMI_CEC_LINE_CLK_DISABLE()  __HAL_RCC_GPIOB_C
#define HDMI_CEC_LINE_AF  GPIO_AF0_CEC
#define HDMI_CEC_IRQn  CEC_CAN_IRQn
#define HDMI_CEC_I2C_ADDRESS  0xA0
```

Define Documentation

#define HDMI_CEC_HPD_SINK_CLK_DISABLE ( ) __HAL_RCC_G

Definition at line 455 of file stm32072b_eval.h.

#define HDMI_CEC_HPD_SINK_CLK_ENABLE ( ) __HAL_RCC_GI

Definition at line 454 of file stm32072b_eval.h.

Referenced by HDMI_CEC_IO_Init().

#define HDMI_CEC_HPD_SINK_GPIO_PORT GPIOD

Definition at line 453 of file stm32072b_eval.h.

Referenced by HDMI_CEC_IO_Init().

#define HDMI_CEC_HPD_SINK_PIN GPIO_PIN_15 /* PD.15 */

I2C HDMI CEC Interface pins.

Definition at line 452 of file stm32072b_eval.h.

Referenced by HDMI_CEC_IO_Init().

#define HDMI_CEC_HPD_SOURCE_CLK_DISABLE ( ) __HAL_RC

Definition at line 460 of file stm32072b_eval.h.

#define HDMI_CEC_HPD_SOURCE_CLK_ENABLE ( ) __HAL_RC

Definition at line 459 of file stm32072b_eval.h.

Referenced by HDMI_CEC_IO_Init().

#define HDMI_CEC_HPD_SOURCE_GPIO_PORT GPIOE

Definition at line 458 of file stm32072b_eval.h.

Referenced by HDMI_CEC_IO_Init().

#define HDMI_CEC_HPD_SOURCE_PIN GPIO_PIN_0 /* PE.0 */

Definition at line 457 of file stm32072b_eval.h.

Referenced by HDMI_CEC_IO_Init().

#define HDMI_CEC_I2C_ADDRESS 0xA0

Definition at line 470 of file stm32072b_eval.h.

#define HDMI_CEC_IRQn CEC_CAN_IRQn

Definition at line 467 of file stm32072b_eval.h.

Referenced by HDMI_CEC_IO_Init().

#define HDMI_CEC_LINE_AF GPIO_AF0_CEC

Definition at line 466 of file stm32072b_eval.h.

Referenced by HDMI_CEC_IO_Init().

#define HDMI_CEC_LINE_CLK_DISABLE ( ) __HAL_RCC_GPIOB_

Definition at line 465 of file stm32072b_eval.h.

#define HDMI_CEC_LINE_CLK_ENABLE ( ) __HAL_RCC_GPIOB_

Definition at line 464 of file stm32072b_eval.h.

Referenced by HDMI_CEC_IO_Init().

#define HDMI_CEC_LINE_GPIO_PORT GPIOB

Definition at line 463 of file stm32072b_eval.h.

Referenced by HDMI_CEC_IO_Init().

#define HDMI_CEC_LINE_PIN GPIO_PIN_8 /* PB.8 */

Definition at line 462 of file stm32072b_eval.h.

Referenced by HDMI_CEC_IO_Init().

#define LCD_CS_HIGH ( ) HAL_GPIO_WritePin(LCD_NCS_GPIO_

Definition at line 416 of file stm32072b_eval.h.

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

#define LCD_CS_LOW ( ) HAL_GPIO_WritePin(LCD_NCS_GPIO_I

Definition at line 415 of file stm32072b_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 423 of file stm32072b_eval.h.

#define LCD_NCS_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOE_(

Definition at line 422 of file stm32072b_eval.h.

Referenced by LCD_IO_Init().

#define LCD_NCS_GPIO_PORT GPIOE /* GPIOE */

Definition at line 421 of file stm32072b_eval.h.

Referenced by LCD_IO_Init(), and SD_IO_Init().

#define LCD_NCS_PIN GPIO_PIN_6 /* PE. 06*/

LCD Control pins.

Definition at line 420 of file stm32072b_eval.h.

Referenced by LCD_IO_Init(), and SD_IO_Init().

#define SD_CS_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOF_CL

Definition at line 436 of file stm32072b_eval.h.

#define SD_CS_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOF_CLK

Definition at line 435 of file stm32072b_eval.h.

Referenced by SD_IO_Init().

#define SD_CS_GPIO_PORT GPIOF /* GPIOF */

Definition at line 434 of file stm32072b_eval.h.

Referenced by SD_IO_Init().

#define SD_CS_HIGH ( ) HAL_GPIO_WritePin(SD_CS_GPIO_POR

Definition at line 429 of file stm32072b_eval.h.

Referenced by SD_IO_CSState(), and SD_IO_Init().

#define SD_CS_LOW ( ) HAL_GPIO_WritePin(SD_CS_GPIO_PORT

Definition at line 428 of file stm32072b_eval.h.

Referenced by SD_IO_CSState().

#define SD_CS_PIN GPIO_PIN_2 /* PF.2 */

SD card Control pin.

Definition at line 433 of file stm32072b_eval.h.

Referenced by SD_IO_Init().

#define SD_DETECT_EXTI_IRQn EXTI4_15_IRQn

Definition at line 445 of file stm32072b_eval.h.

Referenced by SD_IO_Init().

#define SD_DETECT_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIO

Definition at line 444 of file stm32072b_eval.h.

#define SD_DETECT_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOB

Definition at line 443 of file stm32072b_eval.h.

Referenced by SD_IO_Init().

#define SD_DETECT_GPIO_PORT GPIOB /* GPIOB */

Definition at line 442 of file stm32072b_eval.h.

Referenced by BSP_SD_IsDetected(), and SD_IO_Init().

#define SD_DETECT_PIN GPIO_PIN_15 /* PB.15 */

SD Detect Interface pins.

Definition at line 441 of file stm32072b_eval.h.

Referenced by BSP_SD_IsDetected(), and SD_IO_Init().

STM32072B_EVAL BSP User Manual

Functions

BUS Operations Functions

STM32072B_EVAL Common

Functions

static void	**I2C1_Init** (void) I2C Bus initialization.
static void	**I2C1_Error** (void) Manages error callback by re-initializing I2C.
static void	**I2C1_MspInit** (I2C_HandleTypeDef *hi2c) I2C MSP Initialization.
static HAL_StatusTypeDef	**I2C1_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 HAL_StatusTypeDef	**I2C1_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	**I2C1_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.
static HAL_StatusTypeDef	**I2C1_TransmitData** (uint8_t *pBuffer, uint16_t Length) Write buffer through I2C.
static void	**I2C2_Init** (void) I2C Bus initialization.
static void	**I2C2_Error** (void) Discovery I2C2 error treatment function.
static void	**I2C2_MspInit** (I2C_HandleTypeDef *hi2c) I2C MSP Initialization.
static HAL_StatusTypeDef	**I2C2_ReceiveData** (uint16_t Addr, uint8_t *pBuffer, uint16_t Length)

	Read a register of the device through I2C.
static void	**SPIx_Init** (void) SPIx Bus initialization.
static void	**SPIx_Write** (uint8_t Value) SPI Write a byte to device.
static void	**SPIx_WriteReadData** (const uint8_t *DataIn, uint8_t *DataOut, uint16_t DataLegnth) SPI Write a byte to device.
static void	**SPIx_FlushFifo** (void) SPIx_FlushFifo.
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) SPI MSP Init.

Function Documentation

static void I2C1_Error (void) [static]

Manages error callback by re-initializing I2C.

Return values:
None

Definition at line **635** of file **stm32072b_eval.c**.

References **heval_I2c1**, and **I2C1_Init()**.

Referenced by **I2C1_ReadBuffer()**, **I2C1_TransmitData()**, and **I2C1_WriteBuffer()**.

static void I2C1_Init (void) [static]

I2C Bus initialization.

Return values:
None

Definition at line **529** of file **stm32072b_eval.c**.

References **EVAL_I2C1**, **heval_I2c1**, **I2C1_MspInit()**, and **I2C1_TIMING**.

Referenced by **EEPROM_IO_Init()**, **HDMI_CEC_IO_Init()**, **I2C1_Error()**, and **TSENSOR_IO_Init()**.

static HAL_StatusTypeDef I2C1_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 **580** of file **stm32072b_eval.c**.

References **heval_I2c1**, and **I2c1Timeout**.

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

static void I2C1_MspInit (I2C_HandleTypeDef * hi2c) [static]

I2C MSP Initialization.

Parameters:

hi2c	I2C handle

Return values:

None	

Definition at line **649** of file **stm32072b_eval.c**.

References **EVAL_I2C1_CLK_ENABLE**, **EVAL_I2C1_FORCE_RESET**, **EVAL_I2C1_GPIO_CLK_ENABLE**, **EVAL_I2C1_GPIO_PORT**, **EVAL_I2C1_RELEASE_RESET**, **EVAL_I2C1_SCL_PIN**, **EVAL_I2C1_SCL_SDA_AF**, and **EVAL_I2C1_SDA_PIN**.

Referenced by I2C1_Init().

static HAL_StatusTypeDef I2C1_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 **558** of file **stm32072b_eval.c**.

References **heval_I2c1**, **I2C1_Error()**, and **I2c1Timeout**.

Referenced by **EEPROM_IO_ReadData()**, and **TSENSOR_IO_Read()**.

static HAL_StatusTypeDef I2C1_TransmitData (uint8_t * pBuffer, uint16_t Length) [static]

Write buffer through I2C.

Parameters:

pBuffer	The address of the data to be written
Length	buffer size to be written

Return values:

None

Definition at line **615** of file **stm32072b_eval.c**.

References **heval_I2c1**, **I2C1_Error()**, and **I2c1Timeout**.

Referenced by **HDMI_CEC_IO_WriteData()**.

static HAL_StatusTypeDef I2C1_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 **594** of file **stm32072b_eval.c**.

References **heval_I2c1**, **I2C1_Error()**, and **I2c1Timeout**.

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

static void I2C2_Error (void) [static]

Discovery I2C2 error treatment function.

Return values:

None

Definition at line **733** of file **stm32072b_eval.c**.

References **heval_I2c2**, and **I2C2_Init()**.

Referenced by **I2C2_ReceiveData()**.

static void I2C2_Init (void) [static]

I2C Bus initialization.

Return values:

None

Definition at line **687** of file **stm32072b_eval.c**.

References **EVAL_I2C2**, **heval_I2c2**, **I2C2_MspInit()**, and **I2C2_TIMING**.

Referenced by **HDMI_CEC_IO_Init()**, and **I2C2_Error()**.

static void I2C2_MspInit (I2C_HandleTypeDef * hi2c) [static]

I2C MSP Initialization.

Parameters:

hi2c I2C handle

Return values:

None

Definition at line **747** of file **stm32072b_eval.c**.

References **EVAL_I2C2_AF**, **EVAL_I2C2_CLK_ENABLE**, **EVAL_I2C2_FORCE_RESET**, **EVAL_I2C2_GPIO_CLK_ENABLE**, **EVAL_I2C2_GPIO_PORT**, **EVAL_I2C2_RELEASE_RESET**, **EVAL_I2C2_SCL_PIN**, and **EVAL_I2C2_SDA_PIN**.

Referenced by **I2C2_Init()**.

static HAL_StatusTypeDef I2C2_ReceiveData (uint16_t Addr, uint8_t * pBuffer, uint16_t Length) [static]

Read a register of the device through I2C.

Parameters:

Addr Device address on I2C Bus.

pBuffer The address to store the read data

Length buffer size to be read

Return values:

None

Definition at line **714** of file **stm32072b_eval.c**.

References **heval_I2c2**, **I2C2_Error()**, and **I2c2Timeout**.

Referenced by **HDMI_CEC_IO_ReadData()**.

static void SPIx_Error (void) [static]

SPI error treatment function.

Return values:
None

Definition at line **884** of file **stm32072b_eval.c**.

References **heval_Spi**, and **SPIx_Init()**.

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

static void SPIx_FlushFifo (void) [static]

SPIx_FlushFifo.

Return values:
None

Definition at line **875** of file **stm32072b_eval.c**.

References **heval_Spi**.

Referenced by **LCD_IO_WriteMultipleData()**.

static void SPIx_Init (void) [static]

SPIx Bus initialization.

Return values:
None

Definition at line **779** of file **stm32072b_eval.c**.

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

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

static void SPIx_MspInit (SPI_HandleTypeDef * hspi) [static]

SPI MSP Init.

Parameters:

hspi SPI handle

Return values:

None

Definition at line **898** of file **stm32072b_eval.c**.

References **EVAL_SPIx_CLK_ENABLE**, **EVAL_SPIx_FORCE_RESET**, **EVAL_SPIx_MISO_AF**, **EVAL_SPIx_MISO_GPIO_CLK_ENABLE**, **EVAL_SPIx_MISO_GPIO_PORT**, **EVAL_SPIx_MISO_PIN**, **EVAL_SPIx_MOSI_AF**, **EVAL_SPIx_MOSI_DIR_GPIO_CLK_ENABLE**, **EVAL_SPIx_MOSI_DIR_GPIO_PORT**, **EVAL_SPIx_MOSI_DIR_PIN**, **EVAL_SPIx_MOSI_GPIO_CLK_ENABLE**, **EVAL_SPIx_MOSI_GPIO_PORT**, **EVAL_SPIx_MOSI_PIN**, **EVAL_SPIx_RELEASE_RESET**, **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 **812** of file **stm32072b_eval.c**.

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

Referenced by **LCD_IO_ReadData()**.

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 **856** of file **stm32072b_eval.c**.

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

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

static void SPIx_WriteReadData (const uint8_t * DataIn,
uint8_t * DataOut,
uint16_t DataLegnth
) [static]

SPI Write a byte to device.

Parameters:

DataIn value to be written

DataOut read value

DataLegnth data length

Return values:

None

Definition at line **837** of file **stm32072b_eval.c**.

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

Referenced by **SD_IO_WriteByte()**, and **SD_IO_WriteReadData()**.

STM32072B_EVAL BSP User Manual

Private Functions

STM32072B_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 **838** of file **stm32072b_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 **823** of file **stm32072b_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 **908** of file **stm32072b_eval_lcd.c**.

References **lcd_drv**.

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

STM32072B_EVAL BSP User Manual

Defines

STM32072B_EVAL LED

Exported Constants

Define for STM32072B_EVAL board. More...

Defines

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

Detailed Description

Define for STM32072B_EVAL board.

Define Documentation

#define LED1_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOD_CLK

Definition at line 144 of file stm32072b_eval.h.

#define LED1_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOD_CLK_

Definition at line 143 of file stm32072b_eval.h.

#define LED1_GPIO_PORT GPIOD

Definition at line 142 of file stm32072b_eval.h.

#define LED1_PIN GPIO_PIN_8

Definition at line 141 of file stm32072b_eval.h.

#define LED2_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOD_CLK

Definition at line 149 of file stm32072b_eval.h.

#define LED2_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOD_CLK_

Definition at line 148 of file stm32072b_eval.h.

#define LED2_GPIO_PORT GPIOD

Definition at line 147 of file stm32072b_eval.h.

#define LED2_PIN GPIO_PIN_9

Definition at line 146 of file stm32072b_eval.h.

#define LED3_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOD_CLK

Definition at line 154 of file stm32072b_eval.h.

#define LED3_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOD_CLK_

Definition at line 153 of file stm32072b_eval.h.

#define LED3_GPIO_PORT GPIOD

Definition at line 152 of file stm32072b_eval.h.

#define LED3_PIN GPIO_PIN_10

Definition at line 151 of file stm32072b_eval.h.

#define LED4_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOD_CLK

Definition at line 159 of file stm32072b_eval.h.

#define LED4_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOD_CLK_

Definition at line 158 of file stm32072b_eval.h.

#define LED4_GPIO_PORT GPIOD

Definition at line 157 of file stm32072b_eval.h.

#define LED4_PIN GPIO_PIN_11

Definition at line 156 of file stm32072b_eval.h.

#define LEDn 4

Definition at line 139 of file stm32072b_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 166 of file stm32072b_eval.h.

#define LEDx_GPIO_CLK_ENABLE (__LED__)

Value:

```
do { if((__LED__) == LED1) LED1_GPIO_CLK_ENABLE()
; else \
                                                if((__L
ED__) == LED2) LED2_GPIO_CLK_ENABLE(); else \
                                                if((__L
```

```
ED__) == LED3) LED3_GPIO_CLK_ENABLE(); else \
                                          if((__L
ED__) == LED4) LED4_GPIO_CLK_ENABLE();} while(0)
```

Definition at line **161** of file **stm32072b_eval.h**.

Referenced by **BSP_LED_Init()**.

STM32072B_EVAL BSP User Manual

Private Constants

STM32072B_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 90 of file stm32072b_eval_lcd.c.

#define MAX_WIDTH_FONT 24

Definition at line 91 of file stm32072b_eval_lcd.c.

#define OFFSET_BITMAP 54

Definition at line 92 of file stm32072b_eval_lcd.c.

Referenced by LCD_DrawChar().

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

Definition at line 87 of file stm32072b_eval_lcd.c.

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

Definition at line 88 of file stm32072b_eval_lcd.c.

STM32072B_EVAL BSP User Manual

Private Constants

STM32072B_EVAL SD

Defines

#define	**SD_DUMMY_BYTE** 0xFF
#define	**SD_MAX_FRAME_LENGTH** 17 /* Lenght = 16 + 1 */
#define	**SD_CMD_LENGTH** 6
#define	**SD_MAX_TRY** 100 /* Number of try */
#define	**SD_CSD_STRUCT_V1** 0x2 /* CSD struct version V1 */
#define	**SD_CSD_STRUCT_V2** 0x1 /* CSD struct version V2 */
#define	**SD_TOKEN_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_TOKEN_START_DATA_MULTIPLE_BLOCK_READ** 0xF /* Data token start byte, Start Multiple Block Read */
#define	**SD_TOKEN_START_DATA_SINGLE_BLOCK_WRITE** 0xFE Data token start byte, Start Single Block Write */
#define	**SD_TOKEN_START_DATA_MULTIPLE_BLOCK_WRITE** 0x /* Data token start byte, Start Multiple Block Write */
#define	**SD_TOKEN_STOP_DATA_MULTIPLE_BLOCK_WRITE** 0xF /* Data toke stop byte, Stop Multiple Block Write */
#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_IF_COND** 8 /* CMD8 = 0x48 */
#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 SD_CMD_SD_APP_OP_COND 41 /* CMD41 = 0x69 */
#define SD_CMD_APP_CMD 55 /* CMD55 = 0x77 */
#define SD_CMD_READ_OCR 58 /* CMD55 = 0x79 */
```

Enumerations

enum	**SD_Answer_type** {
	SD_ANSWER_R1_EXPECTED,
	SD_ANSWER_R1B_EXPECTED,
	SD_ANSWER_R2_EXPECTED,
	SD_ANSWER_R3_EXPECTED,
	SD_ANSWER_R4R5_EXPECTED,
	SD_ANSWER_R7_EXPECTED
	}
	SD ansewer format. More...

enum	**SD_Error** {
	SD_R1_NO_ERROR = (0x00), **SD_R1_IN_IDLE_STATE** = (0x01), **SD_R1_ERASE_RESET** = (0x02),
	SD_R1_ILLEGAL_COMMAND = (0x04),
	SD_R1_COM_CRC_ERROR = (0x08),
	SD_R1_ERASE_SEQUENCE_ERROR = (0x10),
	SD_R1_ADDRESS_ERROR = (0x20),
	SD_R1_PARAMETER_ERROR = (0x40),
	SD_R2_NO_ERROR = 0x00, **SD_R2_CARD_LOCKED** = 0x01, **SD_R2_LOCKUNLOCK_ERROR** = 0x02,
	SD_R2_ERROR = 0x04,
	SD_R2_CC_ERROR = 0x08, **SD_R2_CARD_ECC_FAILED** = 0x10, **SD_R2_WP_VIOLATION** = 0x20,
	SD_R2_ERASE_PARAM = 0x40,
	SD_R2_OUTOFRANGE = 0x80, **SD_DATA_OK** = (0x05),
	SD_DATA_CRC_ERROR = (0x0B),
	SD_DATA_WRITE_ERROR = (0x0D),
	SD_DATA_OTHER_ERROR = (0xFF)
	}
	SD reponses and error flags. More...

Define Documentation

#define SD_CMD_APP_CMD 55 /* CMD55 = 0x77 */

Definition at line 189 of file stm32072b_eval_sd.c.

Referenced by SD_GoIdleState().

#define SD_CMD_CLR_WRITE_PROT 29 /* CMD29 = 0x5D */

Definition at line 179 of file stm32072b_eval_sd.c.

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

Definition at line 187 of file stm32072b_eval_sd.c.

Referenced by BSP_SD_Erase().

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

Definition at line 185 of file stm32072b_eval_sd.c.

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

Definition at line 184 of file stm32072b_eval_sd.c.

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

Commands: CMDxx = CMD-number | 0x40.

Definition at line 164 of file stm32072b_eval_sd.c.

Referenced by **SD_GoIdleState()**.

#define SD_CMD_LENGTH 6

Definition at line **130** of file **stm32072b_eval_sd.c**.

Referenced by **SD_SendCmd()**.

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

Definition at line **177** of file **stm32072b_eval_sd.c**.

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

Definition at line **173** of file **stm32072b_eval_sd.c**.

#define SD_CMD_READ_OCR 58 /* CMD55 = 0x79 */

Definition at line **190** of file **stm32072b_eval_sd.c**.

Referenced by **SD_GoIdleState()**.

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

Definition at line **172** of file **stm32072b_eval_sd.c**.

Referenced by **BSP_SD_ReadBlocks()**.

#define SD_CMD_SD_APP_OP_COND 41 /* CMD41 = 0x69 */

Definition at line **188** of file **stm32072b_eval_sd.c**.

Referenced by **SD_GoIdleState()**.

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

Definition at line **182** of file **stm32072b_eval_sd.c**.

Referenced by **BSP_SD_Erase()**.

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

Definition at line **181** of file **stm32072b_eval_sd.c**.

Referenced by **BSP_SD_Erase()**.

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

Definition at line **168** of file **stm32072b_eval_sd.c**.

Referenced by **SD_GetCIDRegister()**.

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

Definition at line **167** of file **stm32072b_eval_sd.c**.

Referenced by **SD_GetCSDRegister()**.

#define SD_CMD_SEND_IF_COND 8 /* CMD8 = 0x48 */

Definition at line **166** of file **stm32072b_eval_sd.c**.

Referenced by **SD_GoIdleState()**.

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

Definition at line **165** of file **stm32072b_eval_sd.c**.

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

Definition at line **170** of file **stm32072b_eval_sd.c**.

Referenced by **BSP_SD_GetStatus()**.

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

Definition at line **180** of file **stm32072b_eval_sd.c**.

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

Definition at line **174** of file **stm32072b_eval_sd.c**.

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

Definition at line **171** of file **stm32072b_eval_sd.c**.

Referenced by **BSP_SD_ReadBlocks()**, and **BSP_SD_WriteBlocks()**.

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

Definition at line **178** of file **stm32072b_eval_sd.c**.

#define SD_CMD_STOP_TRANSMISSION 12 /* CMD12 = 0x4C */

Definition at line **169** of file **stm32072b_eval_sd.c**.

#define SD_CMD_UNTAG_ERASE_GROUP 37 /* CMD37 = 0x65 */

Definition at line **186** of file **stm32072b_eval_sd.c**.

#define SD_CMD_UNTAG_SECTOR 34 /* CMD34 = 0x62 */

Definition at line **183** of file **stm32072b_eval_sd.c**.

#define SD_CMD_WRITE_MULT_BLOCK 25 /* CMD25 = 0x59 */

Definition at line **176** of file **stm32072b_eval_sd.c**.

#define SD_CMD_WRITE_SINGLE_BLOCK 24 /* CMD24 = 0x58 */

Definition at line **175** of file **stm32072b_eval_sd.c**.

Referenced by **BSP_SD_WriteBlocks()**.

#define SD_CSD_STRUCT_V1 0x2 /* CSD struct version V1 */

Definition at line **134** of file **stm32072b_eval_sd.c**.

#define SD_CSD_STRUCT_V2 0x1 /* CSD struct version V2 */

Definition at line **135** of file **stm32072b_eval_sd.c**.

#define SD_DUMMY_BYTE 0xFF

Definition at line 127 of file stm32072b_eval_sd.c.

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

#define SD_MAX_FRAME_LENGTH 17 /* Lenght = 16 + 1 */

Definition at line 129 of file stm32072b_eval_sd.c.

#define SD_MAX_TRY 100 /* Number of try */

Definition at line 132 of file stm32072b_eval_sd.c.

Referenced by SD_GoIdleState().

#define SD_TOKEN_START_DATA_MULTIPLE_BLOCK_READ 0xF

Definition at line 156 of file stm32072b_eval_sd.c.

#define SD_TOKEN_START_DATA_MULTIPLE_BLOCK_WRITE 0x

Definition at line 158 of file stm32072b_eval_sd.c.

#define SD_TOKEN_START_DATA_SINGLE_BLOCK_READ 0xFE /

Start Data tokens: Tokens (necessary because at nop/idle (and CS active) only 0xff is on the data/command line)

Definition at line **155** of file **stm32072b_eval_sd.c**.

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

#define SD_TOKEN_START_DATA_SINGLE_BLOCK_WRITE 0xFE

Definition at line **157** of file **stm32072b_eval_sd.c**.

Referenced by **BSP_SD_WriteBlocks()**.

#define SD_TOKEN_STOP_DATA_MULTIPLE_BLOCK_WRITE 0xF

Definition at line **159** of file **stm32072b_eval_sd.c**.

Enumeration Type Documentation

enum SD_Answer_type

SD ansewer format.

Enumerator:

SD_ANSWER_R1_EXPECTED
SD_ANSWER_R1B_EXPECTED
SD_ANSWER_R2_EXPECTED
SD_ANSWER_R3_EXPECTED
SD_ANSWER_R4R5_EXPECTED
SD_ANSWER_R7_EXPECTED

Definition at line **141** of file **stm32072b_eval_sd.c**.

enum SD_Error

SD reponses and error flags.

Enumerator:

SD_R1_NO_ERROR
SD_R1_IN_IDLE_STATE
SD_R1_ERASE_RESET
SD_R1_ILLEGAL_COMMAND
SD_R1_COM_CRC_ERROR
SD_R1_ERASE_SEQUENCE_ERROR
SD_R1_ADDRESS_ERROR
SD_R1_PARAMETER_ERROR
SD_R2_NO_ERROR
SD_R2_CARD_LOCKED
SD_R2_LOCKUNLOCK_ERROR

SD_R2_ERROR
SD_R2_CC_ERROR
SD_R2_CARD_ECC_FAILED
SD_R2_WP_VIOLATION
SD_R2_ERASE_PARAM
SD_R2_OUTOFRANGE

SD_DATA_OK Data response error.

SD_DATA_CRC_ERROR
SD_DATA_WRITE_ERROR
SD_DATA_OTHER_ERROR

Definition at line **195** of file **stm32072b_eval_sd.c**.

STM32072B_EVAL BSP User Manual

Defines

Exported Constants

STM32072B_EVAL SD

Defines

#define	**SD_BLOCK_SIZE** 0x200 Block Size.
#define	**SD_PRESENT** ((uint8_t)0x01) SD detection on its memory slot.
#define	**SD_NOT_PRESENT** ((uint8_t)0x00)

Define Documentation

#define SD_BLOCK_SIZE 0x200

Block Size.

Definition at line 177 of file stm32072b_eval_sd.h.

#define SD_NOT_PRESENT ((uint8_t)0x00)

Definition at line 183 of file stm32072b_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 182 of file stm32072b_eval_sd.h.

Referenced by BSP_SD_Init(), and BSP_SD_IsDetected().

STM32072B_EVAL BSP User Manual

Private Functions

STM32072B_EVAL SD

Functions

static uint8_t	**SD_GetCIDRegister** (**SD_CID** *Cid) Reads the SD card CID register.
static uint8_t	**SD_GetCSDRegister** (**SD_CSD** *Csd) Reads the SD card SCD register.
static uint8_t	**SD_GetDataResponse** (void) Gets the SD card data response and check the busy flag.
static uint8_t	**SD_GoIdleState** (void) Put the SD in Idle state.
static **SD_CmdAnswer_typedef**	**SD_SendCmd** (uint8_t Cmd, uint32_t Arg, uint8_t Crc, uint8_t Answer) Send 5 bytes command to the SD card and get response.
static uint8_t	**SD_WaitData** (uint8_t data) Waits a data from the SD card.
static uint8_t	**SD_ReadData** (void) Waits a data until a value different from SD_DUMMY_BITE.

Function Documentation

uint8_t SD_GetCIDRegister (SD_CID * Cid) [static]

Reads the SD card CID 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 **682** of file **stm32072b_eval_sd.c**.

References **BSP_SD_ERROR**, **BSP_SD_OK**, **SD_CID::CID_CRC**, **SD_CID::ManufactDate**, **SD_CID::ManufacturerID**, **SD_CID::OEM_AppliID**, **SD_CID::ProdName1**, **SD_CID::ProdName2**, **SD_CID::ProdRev**, **SD_CID::ProdSN**, **SD_CmdAnswer_typedef::r1**, **SD_CID::Reserved1**, **SD_CID::Reserved2**, **SD_ANSWER_R1_EXPECTED**, **SD_CMD_SEND_CID**, **SD_DUMMY_BYTE**, **SD_IO_CSState()**, **SD_IO_WriteByte()**, **SD_R1_NO_ERROR**, **SD_SendCmd()**, **SD_TOKEN_START_DATA_SINGLE_BLOCK_READ**, and **SD_WaitData()**.

Referenced by **BSP_SD_GetCardInfo()**.

uint8_t SD_GetCSDRegister (SD_CSD * Csd) [static]

Reads the SD card SCD 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

Definition at line **568** of file **stm32072b_eval_sd.c**.

References **BSP_SD_ERROR**, **BSP_SD_OK**, **SD_CSD::CardComdClasses**, **SD_CSD::CopyFlag**, **SD_CSD::crc**, **SD_CSD::CSDStruct**, **struct_v1::DeviceSize**, **struct_v2::DeviceSize**, **struct_v1::DeviceSizeMul**, **SD_CSD::DSRImpl**, **SD_CSD::EraseSectorSize**, **SD_CSD::EraseSingleBlockEnable**, **SD_CSD::FileFormat**, **SD_CSD::FileFormatGrouop**, **SD_CSD::MaxBusClkFrec**, **struct_v1::MaxRdCurrentVDDMax**, **struct_v1::MaxRdCurrentVDDMin**, **SD_CSD::MaxWrBlockLen**, **struct_v1::MaxWrCurrentVDDMax**, **struct_v1::MaxWrCurrentVDDMin**, **SD_CSD::NSAC**, **SD_CSD::PartBlockRead**, **SD_CSD::PermWrProtect**, **SD_CmdAnswer_typedef::r1**, **SD_CSD::RdBlockLen**, **SD_CSD::RdBlockMisalign**, **struct_v1::Reserved1**, **struct_v2::Reserved1**, **SD_CSD::Reserved1**, **struct_v2::Reserved2**, **SD_CSD::Reserved2**, **SD_CSD::Reserved3**, **SD_CSD::Reserved4**, **SD_CSD::Reserved5**, **SD_ANSWER_R1_EXPECTED**, **SD_CMD_SEND_CSD**, **SD_DUMMY_BYTE**, **SD_IO_CSState()**, **SD_IO_WriteByte()**, **SD_R1_NO_ERROR**, **SD_SendCmd()**, **SD_TOKEN_START_DATA_SINGLE_BLOCK_READ**, **SD_WaitData()**, **SD_CSD::TAAC**, **SD_CSD::TempWrProtect**, **SD_CSD::csd_version::v1**, **SD_CSD::csd_version::v2**, **SD_CSD::version**, **SD_CSD::WrBlockMisalign**, **SD_CSD::WriteBlockPartial**, **SD_CSD::WrProtectGrEnable**, **SD_CSD::WrProtectGrSize**, and **SD_CSD::WrSpeedFact**.

Referenced by **BSP_SD_GetCardInfo()**.

uint8_t SD_GetDataResponse (void) [static]

Gets the SD card data response and check the busy flag.

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 **839** of file **stm32072b_eval_sd.c**.

References **SD_DATA_CRC_ERROR**, **SD_DATA_OK**, **SD_DATA_OTHER_ERROR**, **SD_DATA_WRITE_ERROR**, **SD_DUMMY_BYTE**, **SD_IO_CSState()**, and **SD_IO_WriteByte()**.

Referenced by **BSP_SD_WriteBlocks()**.

uint8_t SD_GoIdleState (void) [static]

Put the SD in Idle state.

Return values:

SD status

Definition at line **880** of file **stm32072b_eval_sd.c**.

References **BSP_SD_ERROR**, **BSP_SD_OK**, **SD_CmdAnswer_typedef::r1**, **SD_CmdAnswer_typedef::r2**, **SD_ANSWER_R1_EXPECTED**, **SD_ANSWER_R3_EXPECTED**, **SD_ANSWER_R7_EXPECTED**, **SD_CMD_APP_CMD**, **SD_CMD_GO_IDLE_STATE**, **SD_CMD_READ_OCR**, **SD_CMD_SD_APP_OP_COND**, **SD_CMD_SEND_IF_COND**,

SD_DUMMY_BYTE, **SD_IO_CSState()**, **SD_IO_WriteByte()**, **SD_MAX_TRY**, **SD_R1_ILLEGAL_COMMAND**, **SD_R1_IN_IDLE_STATE**, **SD_R1_NO_ERROR**, and **SD_SendCmd()**.

Referenced by **BSP_SD_Init()**.

uint8_t SD_ReadData (void) [static]

Waits a data until a value different from SD_DUMMY_BITE.

Return values:

the value read

Definition at line **981** of file **stm32072b_eval_sd.c**.

References **SD_DUMMY_BYTE**, and **SD_IO_WriteByte()**.

Referenced by **SD_SendCmd()**.

SD_CmdAnswer_typedef SD_SendCmd (uint8_t Cmd, uint32_t Arg, uint8_t Crc, uint8_t Answer) [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.
Answer	SD_ANSWER_NOT_EXPECTED or SD_ANSWER_EXPECTED

Return values:

SD status

Definition at line **775** of file **stm32072b_eval_sd.c**.

References **SD_CmdAnswer_typedef::r1**, **SD_CmdAnswer_typedef::r2**, **SD_CmdAnswer_typedef::r3**, **SD_CmdAnswer_typedef::r4**, **SD_CmdAnswer_typedef::r5**, **SD_ANSWER_R1_EXPECTED**, **SD_ANSWER_R1B_EXPECTED**, **SD_ANSWER_R2_EXPECTED**, **SD_ANSWER_R3_EXPECTED**, **SD_ANSWER_R7_EXPECTED**, **SD_CMD_LENGTH**, **SD_DUMMY_BYTE**, **SD_IO_CSState()**, **SD_IO_WriteByte()**, **SD_IO_WriteReadData()**, and **SD_ReadData()**.

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

uint8_t SD_WaitData (uint8_t data) [static]

Waits a data from the SD card.

Parameters:

data Expected data from the SD card

Return values:

BSP_SD_OK or BSP_SD_TIMEOUT

Definition at line **1002** of file **stm32072b_eval_sd.c**.

References **BSP_SD_OK**, **BSP_SD_TIMEOUT**, **SD_DUMMY_BYTE**, and **SD_IO_WriteByte()**.

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

Generated on Wed Jul 5 2017 08:56:11 for STM32072B_EVAL BSP
User Manual by doxygen 1.7.6.1

STM32072B_EVAL BSP User Manual

Private Variables

STM32072B_EVAL TSENSOR

Variables

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

Variable Documentation

TSENSOR_DrvTypeDef* tsensor_drv [static]

Definition at line 80 of file stm32072b_eval_tsensor.c.

Referenced by BSP_TSENSOR_Init(), BSP_TSENSOR_ReadStatus(), and BSP_TSENSOR_ReadTemp().

__IO uint16_t TSENSORAddress = 0

Definition at line 81 of file stm32072b_eval_tsensor.c.

Referenced by BSP_TSENSOR_Init(), BSP_TSENSOR_ReadStatus(), and BSP_TSENSOR_ReadTemp().

STM32072B_EVAL BSP User Manual

Exported Types

STM32072B_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 **68** of file **stm32072b_eval_tsensor.h**.

STM32072B_EVAL BSP User Manual

Defines

Exported Constants

STM32072B_EVAL TSENSOR

Defines

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

Define Documentation

#define TSENSOR_I2C_ADDRESS_A01 0x90

Definition at line 82 of file stm32072b_eval_tsensor.h.

Referenced by BSP_TSENSOR_Init().

#define TSENSOR_I2C_ADDRESS_A02 0x92

Definition at line 83 of file stm32072b_eval_tsensor.h.

Referenced by BSP_TSENSOR_Init().

#define TSENSOR_MAX_TRIALS 50

Definition at line 86 of file stm32072b_eval_tsensor.h.

Referenced by BSP_TSENSOR_Init().

STM32072B_EVAL BSP User Manual

Firmware Directory Reference

Directories

directory	**Drivers**

STM32072B_EVAL BSP User Manual

Drivers Directory Reference

Directories

directory	**BSP**

STM32072B_EVAL BSP User Manual

BSP Directory Reference

Directories

directory	**STM32072B_EVAL**

STM32072B_EVAL BSP User Manual

STM32072B_EVAL Directory Reference

Files

file **stm32072b_eval.c** [code]

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

file **stm32072b_eval.h** [code]

This file contains definitions for STM32072B_EVAL's Leds, push-buttons and COM port hardware resources.

file **stm32072b_eval_eeprom.c** [code]

This file provides a set of functions needed to manage a M24LR64 I2C EEPROM memory.

file **stm32072b_eval_eeprom.h** [code]

This file contains all the functions prototypes for the **stm32072b_eval_eeprom.c** firmware driver.

file **stm32072b_eval_lcd.c** [code]

This file includes the driver for Liquid Crystal Display modules mounted on STM32072B-EVAL evaluation board.

file **stm32072b_eval_lcd.h** [code]

This file contains all the functions prototypes for the **stm32072b_eval_lcd.c** driver.

file **stm32072b_eval_sd.c** [code]

This file provides a set of functions needed to manage the SPI SD Card memory mounted on STM32072B-EVAL board. It implements a high level communication layer for read and write from/to this memory. The needed STM32F0xx hardware resources (SPI and GPIO) are defined in **stm32072b_eval.h** file, and the initialization is performed in **SD_IO_Init()** function declared in **stm32072b_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **SD_IO_Init()** function.

file **stm32072b_eval_sd.h** [code]

This file contains the common defines and functions prototypes for the **stm32072b_eval_sd.c** driver.

file **stm32072b_eval_tsensor.c** [code]

This file provides a set of functions needed to manage the I2C STLM75 temperature sensor mounted on STM32072B-EVAL board . It implements a high level communication layer for read and write from/to this sensor. The needed STM32F0xx hardware resources (I2C and GPIO) are defined in **stm32072b_eval.h** file, and the initialization is performed in **TSENSOR_IO_Init()** function declared in **stm32072b_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 **stm32072b_eval_tsensor.h** [code]

This file contains all the functions prototypes for the **stm32072b_eval_tsensor.c** firmware driver.

Generated on Wed Jul 5 2017 08:56:11 for STM32072B_EVAL BSP
User Manual by doxygen 1.7.6.1

STM32072B_EVAL BSP User Manual

stm32072b_eval.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32072b_eval.h
00004   * @author  MCD Application Team
00005   * @brief   This file contains definitions for STM32072B_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 __STM32072B_EVAL_H
00039 #define __STM32072B_EVAL_H
00040
00041 #ifdef __cplusplus
00042  extern "C" {
00043 #endif
00044
00045 /* Includes ------------------------------
-----------------------------------*/
00046 #include "stm32f0xx_hal.h"
00047
00048 /** @addtogroup BSP
00049   * @{
00050   */
00051
00052 /** @defgroup STM32072B_EVAL STM32072B_EVAL
00053   * @{
00054   */
00055
00056 /** @defgroup STM32072B_EVAL_Common STM32072
B_EVAL Common
00057   * @{
00058   */
00059
00060 /** @defgroup STM32072B_EVAL_Exported_Types
Exported Types
00061   * @{
00062   */
00063
00064 /**
00065  * @brief LED Types Definition
00066  */
```

```
00067 typedef enum
00068 {
00069   LED1 = 0,
00070   LED2 = 1,
00071   LED3 = 2,
00072   LED4 = 3,
00073   /* Color led aliases */
00074   LED_GREEN  = LED1,
00075   LED_ORANGE = LED2,
00076   LED_RED    = LED3,
00077   LED_BLUE   = LED4
00078 }Led_TypeDef;
00079
00080 /**
00081  * @brief BUTTON Types Definition
00082  */
00083 typedef enum
00084 {
00085   BUTTON_TAMPER = 0
00086 }Button_TypeDef;
00087
00088 typedef enum
00089 {
00090   BUTTON_MODE_GPIO = 0,
00091   BUTTON_MODE_EXTI = 1
00092 }ButtonMode_TypeDef;
00093
00094 /**
00095  * @brief JOYSTICK Types Definition
00096  */
00097 typedef enum
00098 {
00099   JOY_SEL   = 0,
00100   JOY_DOWN  = 1,
00101   JOY_LEFT  = 2,
00102   JOY_RIGHT = 3,
00103   JOY_UP    = 4,
```

```
00104   JOY_NONE  = 5
00105 }JOYState_TypeDef;
00106
00107 typedef enum
00108 {
00109   JOY_MODE_GPIO = 0,
00110   JOY_MODE_EXTI = 1
00111 }JOYMode_TypeDef;
00112
00113 /**
00114  * @brief COM Types Definition
00115  */
00116 typedef enum
00117 {
00118   COM1 = 0
00119 }COM_TypeDef;
00120
00121 /**
00122   * @}
00123   */
00124
00125 /** @defgroup STM32072B_EVAL_Exported_Consta
nts Exported Constants
00126   * @{
00127   */
00128
00129 /**
00130   * @brief  Define for STM32072B_EVAL board

00131   */
00132 #if !defined (USE_STM32072B_EVAL)
00133  #define USE_STM32072B_EVAL
00134 #endif
00135
00136 /** @defgroup STM32072B_EVAL_LED STM32072B_E
VAL LED
00137   * @{
```

```
00138   */
00139 #define LEDn                                    4
00140
00141 #define LED1_PIN                                GPI
O_PIN_8
00142 #define LED1_GPIO_PORT                          GPI
OD
00143 #define LED1_GPIO_CLK_ENABLE()                  __H
AL_RCC_GPIOD_CLK_ENABLE()
00144 #define LED1_GPIO_CLK_DISABLE()                 __H
AL_RCC_GPIOD_CLK_DISABLE()
00145
00146 #define LED2_PIN                                GPI
O_PIN_9
00147 #define LED2_GPIO_PORT                          GPI
OD
00148 #define LED2_GPIO_CLK_ENABLE()                  __H
AL_RCC_GPIOD_CLK_ENABLE()
00149 #define LED2_GPIO_CLK_DISABLE()                 __H
AL_RCC_GPIOD_CLK_DISABLE()
00150
00151 #define LED3_PIN                                GPI
O_PIN_10
00152 #define LED3_GPIO_PORT                          GPI
OD
00153 #define LED3_GPIO_CLK_ENABLE()                  __H
AL_RCC_GPIOD_CLK_ENABLE()
00154 #define LED3_GPIO_CLK_DISABLE()                 __H
AL_RCC_GPIOD_CLK_DISABLE()
00155
00156 #define LED4_PIN                                GPI
O_PIN_11
00157 #define LED4_GPIO_PORT                          GPI
OD
00158 #define LED4_GPIO_CLK_ENABLE()                  __H
AL_RCC_GPIOD_CLK_ENABLE()
00159 #define LED4_GPIO_CLK_DISABLE()                 __H
```

```
AL_RCC_GPIOD_CLK_DISABLE()
00160
00161 #define LEDx_GPIO_CLK_ENABLE(__LED__)    do
{ if((__LED__) == LED1) LED1_GPIO_CLK_ENABLE(); el
se \
00162                                              if
((__LED__) == LED2) LED2_GPIO_CLK_ENABLE(); else \
00163                                              if
((__LED__) == LED3) LED3_GPIO_CLK_ENABLE(); else \
00164                                              if
((__LED__) == LED4) LED4_GPIO_CLK_ENABLE();} while
(0)
00165
00166 #define LEDx_GPIO_CLK_DISABLE(__LED__)   (((
__LED__) == LED1) ? LED1_GPIO_CLK_DISABLE() :\
00167                                               ((
__LED__) == LED2) ? LED2_GPIO_CLK_DISABLE() :\
00168                                               ((
__LED__) == LED3) ? LED3_GPIO_CLK_DISABLE() :\
00169                                               ((
__LED__) == LED4) ? LED4_GPIO_CLK_DISABLE() : 0 )
00170
00171 /**
00172   * @}
00173   */
00174
00175 /** @defgroup STM32072B_EVAL_BUTTON STM32072
B_EVAL BUTTON
00176   * @{
00177   */
00178 #define JOYn                                5
00179 #define BUTTONn                             1
00180
00181 /**
00182  * @brief Tamper push-button
00183  */
00184 #define TAMPER_BUTTON_PIN                   GPI
```

```
O_PIN_13
00185 #define TAMPER_BUTTON_GPIO_PORT          GPI
OC
00186 #define TAMPER_BUTTON_GPIO_CLK_ENABLE()  __H
AL_RCC_GPIOC_CLK_ENABLE()
00187 #define TAMPER_BUTTON_GPIO_CLK_DISABLE() __H
AL_RCC_GPIOC_CLK_DISABLE()
00188 #define TAMPER_BUTTON_EXTI_IRQn          EXT
I4_15_IRQn
00189
00190 #define TAMPERx_GPIO_CLK_ENABLE(__BUTTON__)
   do { if((__BUTTON__) == BUTTON_TAMPER) TAMPER_B
UTTON_GPIO_CLK_ENABLE();} while(0)
00191
00192 #define TAMPERx_GPIO_CLK_DISABLE(__BUTTON__)
   (((__BUTTON__) == BUTTON_TAMPER) ? TAMPER_BUTTO
N_GPIO_CLK_DISABLE(): 0 )
00193
00194 /**
00195  * @brief Joystick Right push-button
00196  */
00197 #define RIGHT_JOY_PIN                    GPI
O_PIN_3
00198 #define RIGHT_JOY_GPIO_PORT              GPI
OE
00199 #define RIGHT_JOY_GPIO_CLK_ENABLE()      __H
AL_RCC_GPIOE_CLK_ENABLE()
00200 #define RIGHT_JOY_GPIO_CLK_DISABLE()     __H
AL_RCC_GPIOE_CLK_DISABLE()
00201 #define RIGHT_JOY_EXTI_IRQn              EXT
I2_3_IRQn
00202
00203 /**
00204  * @brief Joystick Left push-button
00205  */
00206 #define LEFT_JOY_PIN                     GPI
O_PIN_2
```

```
00207 #define LEFT_JOY_GPIO_PORT                 GPI
OE
00208 #define LEFT_JOY_GPIO_CLK_ENABLE()         __H
AL_RCC_GPIOE_CLK_ENABLE()
00209 #define LEFT_JOY_GPIO_CLK_DISABLE()        __H
AL_RCC_GPIOE_CLK_DISABLE()
00210 #define LEFT_JOY_EXTI_IRQn                 EXT
I2_3_IRQn
00211
00212 /**
00213  * @brief Joystick Up push-button
00214  */
00215 #define UP_JOY_PIN                         GPI
O_PIN_9
00216 #define UP_JOY_GPIO_PORT                   GPI
OF
00217 #define UP_JOY_GPIO_CLK_ENABLE()           __H
AL_RCC_GPIOF_CLK_ENABLE()
00218 #define UP_JOY_GPIO_CLK_DISABLE()          __H
AL_RCC_GPIOF_CLK_DISABLE()
00219 #define UP_JOY_EXTI_IRQn                   EXT
I4_15_IRQn
00220
00221 /**
00222  * @brief Joystick Down push-button
00223  */
00224 #define DOWN_JOY_PIN                       GPI
O_PIN_10
00225 #define DOWN_JOY_GPIO_PORT                 GPI
OF
00226 #define DOWN_JOY_GPIO_CLK_ENABLE()         __H
AL_RCC_GPIOF_CLK_ENABLE()
00227 #define DOWN_JOY_GPIO_CLK_DISABLE()        __H
AL_RCC_GPIOF_CLK_DISABLE()
00228 #define DOWN_JOY_EXTI_IRQn                 EXT
I4_15_IRQn
00229
```

```
00230 /**
00231  * @brief Joystick Sel push-button
00232  */
00233 #define SEL_JOY_PIN                     GPI
O_PIN_0
00234 #define SEL_JOY_GPIO_PORT               GPI
OA
00235 #define SEL_JOY_GPIO_CLK_ENABLE()       __H
AL_RCC_GPIOA_CLK_ENABLE()
00236 #define SEL_JOY_GPIO_CLK_DISABLE()      __H
AL_RCC_GPIOA_CLK_DISABLE()
00237 #define SEL_JOY_EXTI_IRQn               EXT
I0_1_IRQn
00238
00239 #define JOYx_GPIO_CLK_ENABLE(__JOY__)    do
{ if((__JOY__) == JOY_SEL) SEL_JOY_GPIO_CLK_ENABLE
(); else \
00240                                             if
((__JOY__) == JOY_DOWN) DOWN_JOY_GPIO_CLK_ENABLE()
; else \
00241                                             if
((__JOY__) == JOY_LEFT) LEFT_JOY_GPIO_CLK_ENABLE()
; else \
00242                                             if
((__JOY__) == JOY_RIGHT) RIGHT_JOY_GPIO_CLK_ENABLE
(); else \
00243                                             if
((__JOY__) == JOY_UP) UP_JOY_GPIO_CLK_ENABLE();} w
hile(0)
00244
00245 #define JOYx_GPIO_CLK_DISABLE(__JOY__)   (((
__JOY__) == JOY_SEL) ? SEL_JOY_GPIO_CLK_DISABLE()
:\
00246                                             ((
__JOY__) == JOY_DOWN) ? DOWN_JOY_GPIO_CLK_DISABLE(
) :\
00247                                             ((
```

```
__JOY__) == JOY_LEFT) ? LEFT_JOY_GPIO_CLK_DISABLE(
) :\
00248                                                ((
__JOY__) == JOY_RIGHT) ? RIGHT_JOY_GPIO_CLK_DISABL
E() :\
00249                                                ((
__JOY__) == JOY_UP) ? UP_JOY_GPIO_CLK_DISABLE() :
0 )
00250
00251 /**
00252   * @}
00253   */
00254
00255 /** @defgroup STM32072B_EVAL_COM STM32072B_E
VAL COM
00256   * @{
00257   */
00258 #define COMn                                1
00259
00260 /**
00261  * @brief Definition for COM port1, connecte
d to USART2
00262  */
00263 #define EVAL_COM1                           USAR
T2
00264 #define EVAL_COM1_CLK_ENABLE()              __HA
L_RCC_USART2_CLK_ENABLE()
00265 #define EVAL_COM1_CLK_DISABLE()             __HA
L_RCC_USART2_CLK_DISABLE()
00266
00267 #define EVAL_COM1_TX_PIN                    GPIO
_PIN_5
00268 #define EVAL_COM1_TX_GPIO_PORT              GPIOD

00269 #define EVAL_COM1_TX_GPIO_CLK_ENABLE()  __HA
L_RCC_GPIOD_CLK_ENABLE()
00270 #define EVAL_COM1_TX_GPIO_CLK_DISABLE() __HA
```

```
L_RCC_GPIOD_CLK_DISABLE()
00271 #define EVAL_COM1_TX_AF                  GPIO
_AF0_USART2
00272
00273 #define EVAL_COM1_RX_PIN                 GPIO
_PIN_6
00274 #define EVAL_COM1_RX_GPIO_PORT           GPIOD

00275 #define EVAL_COM1_RX_GPIO_CLK_ENABLE()  __HA
L_RCC_GPIOD_CLK_ENABLE()
00276 #define EVAL_COM1_RX_GPIO_CLK_DISABLE() __HA
L_RCC_GPIOD_CLK_DISABLE()
00277 #define EVAL_COM1_RX_AF                  GPIO
_AF0_USART2
00278
00279 #define EVAL_COM1_CTS_PIN                GPIO
_PIN_3
00280 #define EVAL_COM1_CTS_GPIO_PORT          GPIOD

00281 #define EVAL_COM1_CTS_GPIO_CLK_ENABLE() __HA
L_RCC_GPIOD_CLK_ENABLE()
00282 #define EVAL_COM1_CTS_GPIO_CLK_DISABLE() __H
AL_RCC_GPIOD_CLK_DISABLE()
00283 #define EVAL_COM1_CTS_AF                 GPIO
_AF0_USART2
00284
00285 #define EVAL_COM1_RTS_PIN                GPIO
_PIN_4
00286 #define EVAL_COM1_RTS_GPIO_PORT          GPIOD

00287 #define EVAL_COM1_RTS_GPIO_CLK_ENABLE() __HA
L_RCC_GPIOD_CLK_ENABLE()
00288 #define EVAL_COM1_RTS_GPIO_CLK_DISABLE() __H
AL_RCC_GPIOD_CLK_DISABLE()
00289 #define EVAL_COM1_RTS_AF                 GPIO
_AF0_USART2
00290
```

```
00291 #define EVAL_COM1_IRQn                    USAR
T2_IRQn
00292
00293 #define COMx_CLK_ENABLE(__COM__)          do {
 if((__COM__) == COM1) EVAL_COM1_CLK_ENABLE();} wh
ile(0)
00294 #define COMx_CLK_DISABLE(__COM__)         (((_
_COM__) == COM1) ? EVAL_COM1_CLK_DISABLE() : 0)
00295
00296 #define COMx_TX_GPIO_CLK_ENABLE(__COM__) do
{ if((__COM__) == COM1) EVAL_COM1_TX_GPIO_CLK_ENAB
LE();} while(0)
00297 #define COMx_TX_GPIO_CLK_DISABLE(__COM__) ((
(__COM__) == COM1) ? EVAL_COM1_TX_GPIO_CLK_DISABLE
() : 0)
00298
00299 #define COMx_RX_GPIO_CLK_ENABLE(__COM__) do
{ if((__COM__) == COM1) EVAL_COM1_RX_GPIO_CLK_ENAB
LE();} while(0)
00300 #define COMx_RX_GPIO_CLK_DISABLE(__COM__) ((
(__COM__) == COM1) ? EVAL_COM1_RX_GPIO_CLK_DISABLE
() : 0)
00301
00302 #define COMx_CTS_GPIO_CLK_ENABLE(__COM__) do
 { if((__COM__) == COM1) EVAL_COM1_CTS_GPIO_CLK_EN
ABLE();} while(0)
00303 #define COMx_CTS_GPIO_CLK_DISABLE(__COM__) (
((__COM__) == COM1) ? EVAL_COM1_CTS_GPIO_CLK_DISAB
LE() : 0)
00304
00305 #define COMx_RTS_GPIO_CLK_ENABLE(__COM__) do
 { if((__COM__) == COM1) EVAL_COM1_RTS_GPIO_CLK_EN
ABLE();} while(0)
00306 #define COMx_RTS_GPIO_CLK_DISABLE(__COM__) (
((__COM__) == COM1) ? EVAL_COM1_RTS_GPIO_CLK_DISAB
LE() : 0)
00307 /**
```

```
00308   * @}
00309   */
00310
00311 /** @addtogroup STM32072B_EVAL_BUS STM32072B
_EVAL BUS
00312   * @{
00313   */
00314 #if defined(HAL_I2C_MODULE_ENABLED)
00315 /*##################### I2C2 ###############
#####################*/
00316 /* User can use this section to tailor I2Cx
instance used and associated resources */
00317 /* Definition for I2C1 Pins */
00318 #define EVAL_I2C1                          I2C1

00319 #define EVAL_I2C1_CLK_ENABLE()             __H
AL_RCC_I2C1_CLK_ENABLE()
00320 #define EVAL_I2C1_CLK_DISABLE()            __H
AL_RCC_I2C1_CLK_DISABLE()
00321 #define EVAL_I2C1_FORCE_RESET()            __H
AL_RCC_I2C1_FORCE_RESET()
00322 #define EVAL_I2C1_RELEASE_RESET()          __H
AL_RCC_I2C1_RELEASE_RESET()
00323
00324 #define EVAL_I2C1_SCL_PIN                  GPI
O_PIN_6             /* PB.6 */
00325 #define EVAL_I2C1_SDA_PIN                  GPI
O_PIN_7             /* PB.7 */
00326
00327 #define EVAL_I2C1_GPIO_PORT                GPI
OB                  /* GPIOB */
00328 #define EVAL_I2C1_GPIO_CLK_ENABLE()        __H
AL_RCC_GPIOB_CLK_ENABLE()
00329 #define EVAL_I2C1_GPIO_CLK_DISABLE()       __H
AL_RCC_GPIOB_CLK_DISABLE()
00330 #define EVAL_I2C1_SCL_SDA_AF               GPI
O_AF1_I2C1
```

```
00331
00332 /* Definition for I2C2 Pins */
00333 #define EVAL_I2C2                           I2C2

00334 #define EVAL_I2C2_CLK_ENABLE()              __H
AL_RCC_I2C2_CLK_ENABLE()
00335 #define EVAL_I2C2_CLK_DISABLE()             __H
AL_RCC_I2C2_CLK_DISABLE()
00336 #define EVAL_I2C2_FORCE_RESET()             __H
AL_RCC_I2C2_FORCE_RESET()
00337 #define EVAL_I2C2_RELEASE_RESET()           __H
AL_RCC_I2C2_RELEASE_RESET()
00338
00339 #define EVAL_I2C2_SCL_PIN                   GPI
O_PIN_13            /* PB.13 */
00340 #define EVAL_I2C2_SDA_PIN                   GPI
O_PIN_14            /* PB.14 */
00341
00342 #define EVAL_I2C2_GPIO_PORT                 GPI
OB                   /* GPIOB */
00343 #define EVAL_I2C2_GPIO_CLK_ENABLE()         __H
AL_RCC_GPIOB_CLK_ENABLE()
00344 #define EVAL_I2C2_GPIO_CLK_DISABLE()        __H
AL_RCC_GPIOB_CLK_DISABLE()
00345 #define EVAL_I2C2_AF                        GPI
O_AF5_I2C2
00346
00347 /* Definition for I2C2 NVIC */
00348 #define EVAL_I2C2_IRQn                      I2C
2_IRQn
00349
00350 /* Maximum Timeout values for flags waiting
loops. These timeouts are not based
00351    on accurate values, they just guarantee t
hat the application will not remain
00352    stuck if the I2C communication is corrupt
ed.
```

```
00353    You may modify these timeout values depen
ding on CPU frequency and application
00354    conditions (interrupts routines ...). */

00355 #define EVAL_I2C1_TIMEOUT_MAX            1000

00356 #define EVAL_I2C2_TIMEOUT_MAX            1000

00357
00358 /* I2C TIMING is calculated in case of the I
2C Clock source is the SYSCLK = 48 MHz */
00359 /* Set TIMING to 0x00E0D3FF to reach 100 KHz
 speed (Rise time = 50ns, Fall time = 10ns) */
00360 #define I2C2_TIMING                      0x0
0E0D3FF
00361 #define I2C1_TIMING                      0x0
0E0D3FF
00362
00363 #endif /* HAL_I2C_MODULE_ENABLED */
00364
00365 #if defined(HAL_SPI_MODULE_ENABLED)
00366 /**
00367   * @brief  Definition for SPI Interface pin
s (SPI1 used)
00368   */
00369 #define EVAL_SPIx                        SPI1

00370 #define EVAL_SPIx_CLK_ENABLE()           __H
AL_RCC_SPI1_CLK_ENABLE()
00371 #define EVAL_SPIx_CLK_DISABLE()          __H
AL_RCC_SPI1_CLK_DISABLE()
00372 #define EVAL_SPIx_FORCE_RESET()          __H
AL_RCC_SPI1_FORCE_RESET()
00373 #define EVAL_SPIx_RELEASE_RESET()        __H
AL_RCC_SPI1_RELEASE_RESET()
00374
00375 #define EVAL_SPIx_SCK_PIN                GPI
```

```
O_PIN_3                 /* PB.03 */
00376 #define EVAL_SPIx_SCK_GPIO_PORT          GPI
OB                      /* GPIOB */
00377 #define EVAL_SPIx_SCK_GPIO_CLK_ENABLE()  __H
AL_RCC_GPIOB_CLK_ENABLE()
00378 #define EVAL_SPIx_SCK_GPIO_CLK_DISABLE() __H
AL_RCC_GPIOB_CLK_DISABLE()
00379 #define EVAL_SPIx_SCK_AF                 GPI
O_AF0_SPI1
00380
00381 #define EVAL_SPIx_MISO_PIN               GPI
O_PIN_14                /* PE.14 */
00382 #define EVAL_SPIx_MISO_GPIO_PORT         GPI
OE                      /* GPIOE */
00383 #define EVAL_SPIx_MISO_GPIO_CLK_ENABLE() __H
AL_RCC_GPIOE_CLK_ENABLE()
00384 #define EVAL_SPIx_MISO_GPIO_CLK_DISABLE() __
HAL_RCC_GPIOE_CLK_DISABLE()
00385 #define EVAL_SPIx_MISO_AF                GPI
O_AF1_SPI1
00386
00387 #define EVAL_SPIx_MOSI_PIN               GPI
O_PIN_15                /* PE.15 */
00388 #define EVAL_SPIx_MOSI_GPIO_PORT         GPI
OE                      /* GPIOE */
00389 #define EVAL_SPIx_MOSI_GPIO_CLK_ENABLE() __H
AL_RCC_GPIOE_CLK_ENABLE()
00390 #define EVAL_SPIx_MOSI_GPIO_CLK_DISABLE() __
HAL_RCC_GPIOE_CLK_DISABLE()
00391 #define EVAL_SPIx_MOSI_AF                GPI
O_AF1_SPI1
00392
00393 #define EVAL_SPIx_MOSI_DIR_PIN           GPI
O_PIN_2               /* PB.02 */
00394 #define EVAL_SPIx_MOSI_DIR_GPIO_PORT     GPI
OB                      /* GPIOB */
00395 #define EVAL_SPIx_MOSI_DIR_GPIO_CLK_ENABLE()
```

```
 __HAL_RCC_GPIOB_CLK_ENABLE()
00396 #define EVAL_SPIx_MOSI_DIR_GPIO_CLK_DISABLE(
) __HAL_RCC_GPIOB_CLK_DISABLE()
00397
00398 /* Maximum Timeout values for flags waiting
loops. These timeouts are not based
00399    on accurate values, they just guarantee t
hat the application will not remain
00400    stuck if the SPI communication is corrupt
ed.
00401    You may modify these timeout values depen
ding on CPU frequency and application
00402    conditions (interrupts routines ...). */

00403 #define EVAL_SPIx_TIMEOUT_MAX
  1000
00404
00405 #endif /* HAL_SPI_MODULE_ENABLED */
00406 /**
00407   * @}
00408   */
00409
00410 /** @defgroup STM32072B_EVAL_COMPONENT STM32
072B_EVAL COMPONENT
00411   * @{
00412   */
00413 /*####################### LCD ###############
#####################*/
00414 /* Chip Select macro definition */
00415 #define LCD_CS_LOW()                    HAL_
GPIO_WritePin(LCD_NCS_GPIO_PORT, LCD_NCS_PIN, GPIO
_PIN_RESET)
00416 #define LCD_CS_HIGH()                   HAL_
GPIO_WritePin(LCD_NCS_GPIO_PORT, LCD_NCS_PIN, GPIO
_PIN_SET)
00417 /**
00418   * @brief  LCD Control pins
```

```
00419   */
00420 #define LCD_NCS_PIN                     GPIO
_PIN_6              /* PE. 06*/
00421 #define LCD_NCS_GPIO_PORT               GPIO
E                   /* GPIOE */
00422 #define LCD_NCS_GPIO_CLK_ENABLE()       __HA
L_RCC_GPIOE_CLK_ENABLE()
00423 #define LCD_NCS_GPIO_CLK_DISABLE()      __HA
L_RCC_GPIOE_CLK_DISABLE()
00424
00425
00426 /*###################### SD ################
####################*/
00427 /* Chip Select macro definition */
00428 #define SD_CS_LOW()                     HAL_
GPIO_WritePin(SD_CS_GPIO_PORT, SD_CS_PIN, GPIO_PIN
_RESET)
00429 #define SD_CS_HIGH()                    HAL_
GPIO_WritePin(SD_CS_GPIO_PORT, SD_CS_PIN, GPIO_PIN
_SET)
00430 /**
00431   * @brief SD card Control pin
00432   */
00433 #define SD_CS_PIN                       GPIO
_PIN_2              /* PF.2 */
00434 #define SD_CS_GPIO_PORT                 GPIO
F                   /* GPIOF */
00435 #define SD_CS_GPIO_CLK_ENABLE()         __HA
L_RCC_GPIOF_CLK_ENABLE()
00436 #define SD_CS_GPIO_CLK_DISABLE()        __HA
L_RCC_GPIOF_CLK_DISABLE()
00437
00438 /**
00439   * @brief  SD Detect Interface pins
00440   */
00441 #define SD_DETECT_PIN                   GPIO
_PIN_15             /* PB.15 */
```

```
00442 #define SD_DETECT_GPIO_PORT                GPIO
B                     /* GPIOB */
00443 #define SD_DETECT_GPIO_CLK_ENABLE()     __HA
L_RCC_GPIOB_CLK_ENABLE()
00444 #define SD_DETECT_GPIO_CLK_DISABLE()    __HA
L_RCC_GPIOB_CLK_DISABLE()
00445 #define SD_DETECT_EXTI_IRQn                EXTI
4_15_IRQn
00446
00447
00448 /*###################### HDMI-CEC ###########
#########################*/
00449 /**
00450   * @brief  I2C HDMI CEC Interface pins
00451   */
00452 #define HDMI_CEC_HPD_SINK_PIN              GP
IO_PIN_15  /* PD.15 */
00453 #define HDMI_CEC_HPD_SINK_GPIO_PORT        GP
IOD
00454 #define HDMI_CEC_HPD_SINK_CLK_ENABLE()    __
HAL_RCC_GPIOD_CLK_ENABLE()
00455 #define HDMI_CEC_HPD_SINK_CLK_DISABLE()   __
HAL_RCC_GPIOD_CLK_DISABLE()
00456
00457 #define HDMI_CEC_HPD_SOURCE_PIN            GP
IO_PIN_0  /* PE.0 */
00458 #define HDMI_CEC_HPD_SOURCE_GPIO_PORT      GP
IOE
00459 #define HDMI_CEC_HPD_SOURCE_CLK_ENABLE()  __
HAL_RCC_GPIOE_CLK_ENABLE()
00460 #define HDMI_CEC_HPD_SOURCE_CLK_DISABLE() __
HAL_RCC_GPIOE_CLK_DISABLE()
00461
00462 #define HDMI_CEC_LINE_PIN                  GP
IO_PIN_8  /* PB.8 */
00463 #define HDMI_CEC_LINE_GPIO_PORT            GP
IOB
```

```
00464 #define HDMI_CEC_LINE_CLK_ENABLE()          __
HAL_RCC_GPIOB_CLK_ENABLE()
00465 #define HDMI_CEC_LINE_CLK_DISABLE()         __
HAL_RCC_GPIOB_CLK_DISABLE()
00466 #define HDMI_CEC_LINE_AF                    GP
IO_AF0_CEC
00467 #define HDMI_CEC_IRQn                       CE
C_CAN_IRQn
00468
00469 /* HDMI-CEC hardware I2C address */
00470 #define HDMI_CEC_I2C_ADDRESS                0x
A0
00471
00472 /**
00473   * @}
00474   */
00475
00476 /**
00477   * @}
00478   */
00479
00480 /** @defgroup STM32072B_EVAL_Exported_Functi
ons Exported Functions
00481   * @{
00482   */
00483 uint32_t          BSP_GetVersion(void);
00484 void              BSP_LED_Init(Led_TypeDef L
ed);
00485 void              BSP_LED_On(Led_TypeDef Led
);
00486 void              BSP_LED_Off(Led_TypeDef Le
d);
00487 void              BSP_LED_Toggle(Led_TypeDef
 Led);
00488 void              BSP_PB_Init(Button_TypeDef
 Button, ButtonMode_TypeDef Button_Mode);
00489 uint32_t          BSP_PB_GetState(Button_Typ
```

```
eDef Button);
00490 uint8_t           BSP_JOY_Init(JOYMode_TypeD
ef Joy_Mode);
00491 JOYState_TypeDef  BSP_JOY_GetState(void);
00492 #if defined(HAL_UART_MODULE_ENABLED)
00493 void              BSP_COM_Init(COM_TypeDef C
OM, UART_HandleTypeDef* huart);
00494 #endif /* HAL_UART_MODULE_ENABLED */
00495
00496 /**
00497   * @}
00498   */
00499
00500 /**
00501   * @}
00502   */
00503
00504 /**
00505   * @}
00506   */
00507
00508 /**
00509   * @}
00510   */
00511
00512 #ifdef __cplusplus
00513 }
00514 #endif
00515
00516 #endif /* __STM32072B_EVAL_H */
00517
00518 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32072B_EVAL BSP User Manual

stm32072b_eval.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32072b_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 ports
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 /* Includes ------------------------------
------------------------------------*/
00038 #include "stm32072b_eval.h"
00039
00040 /** @addtogroup BSP
00041   * @{
00042   */
00043
00044 /** @addtogroup STM32072B_EVAL
00045   * @{
00046   */
00047
00048 /** @addtogroup STM32072B_EVAL_Common
00049   * @brief This file provides firmware funct
ions to manage Leds, push-buttons,
00050   *        COM ports, SD card on SPI and tem
perature sensor (LM75) available on
00051   *        STM32072B-EVAL evaluation board f
rom STMicroelectronics.
00052   * @{
00053   */
00054
00055 /** @defgroup STM32072B_EVAL_Private_Constan
ts Private Constants
00056   * @{
00057   */
00058 /* LINK LCD */
00059 #define START_BYTE          0x70
00060 #define SET_INDEX           0x00
00061 #define READ_STATUS         0x01
00062 #define LCD_WRITE_REG       0x02
00063 #define LCD_READ_REG        0x03
00064
00065 /* LINK SD Card */
00066 #define SD_DUMMY_BYTE       0xFF
```

```
00067 #define SD_NO_RESPONSE_EXPECTED  0x80
00068
00069 /**
00070   * @brief STM32072B EVAL BSP Driver version
 number V2.1.8
00071   */
00072 #define __STM32072B_EVAL_BSP_VERSION_MAIN
(0x02) /*!< [31:24] main version */
00073 #define __STM32072B_EVAL_BSP_VERSION_SUB1
(0x01) /*!< [23:16] sub1 version */
00074 #define __STM32072B_EVAL_BSP_VERSION_SUB2
(0x08) /*!< [15:8]  sub2 version */
00075 #define __STM32072B_EVAL_BSP_VERSION_RC
(0x00) /*!< [7:0]  release candidate */
00076 #define __STM32072B_EVAL_BSP_VERSION
 ((__STM32072B_EVAL_BSP_VERSION_MAIN << 24)\
00077
 |(__STM32072B_EVAL_BSP_VERSION_SUB1 << 16)\
00078
 |(__STM32072B_EVAL_BSP_VERSION_SUB2 << 8 )\
00079
 |(__STM32072B_EVAL_BSP_VERSION_RC))
00080 /**
00081   * @}
00082   */
00083
00084 /** @defgroup STM32072B_EVAL_Private_Variabl
es Private Variables
00085   * @{
00086   */
00087 /**
00088   * @brief LED variables
00089   */
00090 GPIO_TypeDef* LED_PORT[LEDn] = {LED1_GPIO_PO
RT,
00091                                 LED2_GPIO_PO
RT,
```

```
00092                                           LED3_GPIO_PO
RT,
00093                                           LED4_GPIO_PO
RT};
00094
00095 const uint16_t LED_PIN[LEDn] = {LED1_PIN,
00096                                           LED2_PIN,
00097                                           LED3_PIN,
00098                                           LED4_PIN};
00099 /**
00100   * @brief BUTTON variables
00101   */
00102 GPIO_TypeDef* BUTTON_PORT[BUTTONn] = {TAMPER
_BUTTON_GPIO_PORT};
00103 const uint16_t BUTTON_PIN[BUTTONn] = {TAMPER
_BUTTON_PIN};
00104 const uint8_t BUTTON_IRQn[BUTTONn] = {TAMPER
_BUTTON_EXTI_IRQn};
00105
00106 /**
00107   * @brief JOYSTICK variables
00108   */
00109 GPIO_TypeDef* JOY_PORT[JOYn] = {SEL_JOY_GPIO
_PORT,
00110                                           DOWN_JOY_GPI
O_PORT,
00111                                           LEFT_JOY_GPI
O_PORT,
00112                                           RIGHT_JOY_GP
IO_PORT,
00113                                           UP_JOY_GPIO_
PORT};
00114
00115 const uint16_t JOY_PIN[JOYn] = {SEL_JOY_PIN,
00116                                           DOWN_JOY_PIN
,
00117                                           LEFT_JOY_PIN
```

```
,
00118                                       RIGHT_JOY_PIN
,
00119                                       UP_JOY_PIN};
00120
00121 const uint8_t JOY_IRQn[JOYn] = {SEL_JOY_EXTI
_IRQn,
00122                                       DOWN_JOY_EXT
I_IRQn,
00123                                       LEFT_JOY_EXT
I_IRQn,
00124                                       RIGHT_JOY_EX
TI_IRQn,
00125                                       UP_JOY_EXTI_
IRQn};
00126
00127 /**
00128   * @brief COM variables
00129   */
00130 #ifdef HAL_UART_MODULE_ENABLED
00131 USART_TypeDef* COM_USART[COMn]  = {EVAL_COM1
};
00132
00133 GPIO_TypeDef* COM_TX_PORT[COMn] = {EVAL_COM1
_TX_GPIO_PORT};
00134
00135 GPIO_TypeDef* COM_RX_PORT[COMn] = {EVAL_COM1
_RX_GPIO_PORT};
00136
00137 const uint16_t COM_TX_PIN[COMn] = {EVAL_COM1
_TX_PIN};
00138
00139 const uint16_t COM_RX_PIN[COMn] = {EVAL_COM1
_RX_PIN};
00140
00141 const uint16_t COM_TX_AF[COMn]  = {EVAL_COM1
_TX_AF};
```

```
00142
00143 const uint16_t COM_RX_AF[COMn]  = {EVAL_COM1
_RX_AF};
00144
00145 #endif /*HAL_UART_MODULE_ENABLED*/
00146
00147 /**
00148  * @brief BUS variables
00149  */
00150 #if defined(HAL_I2C_MODULE_ENABLED)
00151 uint32_t I2c1Timeout = EVAL_I2C1_TIMEOUT_MAX
;    /*<! Value of Timeout when I2C1 communication
 fails */
00152 uint32_t I2c2Timeout = EVAL_I2C2_TIMEOUT_MAX
;    /*<! Value of Timeout when I2C2 communication
 fails */
00153 I2C_HandleTypeDef heval_I2c1;
00154 I2C_HandleTypeDef heval_I2c2;
00155 #endif /* HAL_I2C_MODULE_ENABLED */
00156
00157 #if defined(HAL_SPI_MODULE_ENABLED)
00158 uint32_t SpixTimeout = EVAL_SPIx_TIMEOUT_MAX
;    /*<! Value of Timeout when SPI communication
fails */
00159 static SPI_HandleTypeDef heval_Spi;
00160 #endif /* HAL_SPI_MODULE_ENABLED */
00161
00162 /**
00163   * @}
00164   */
00165
00166 /** @defgroup STM32072B_EVAL_BUS_Operations_
Functions BUS Operations Functions
00167   * @{
00168   */
00169 #if defined(HAL_I2C_MODULE_ENABLED)
00170 /* I2Cx bus function */
```

```
00171 /* Link function for I2C EEPROM, TSENSOR & H
DMI_SOURCE peripherals */
00172 static void                I2C1_Init(void);
00173 static void                I2C1_Error (void);
00174 static void                I2C1_MspInit(I2C_H
andleTypeDef *hi2c);
00175 /* Link function for I2C EEPROM & TSENSOR pe
ripherals */
00176 static HAL_StatusTypeDef  I2C1_WriteBuffer(u
int16_t Addr, uint8_t Reg, uint16_t RegSize, uint8
_t *pBuffer, uint16_t Length);
00177 static HAL_StatusTypeDef  I2C1_ReadBuffer(ui
nt16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_
t *pBuffer, uint16_t Length);
00178 static HAL_StatusTypeDef  I2C1_IsDeviceReady
(uint16_t DevAddress, uint32_t Trials);
00179 /* Link function for HDMI_SOURCE peripheral
*/
00180 static HAL_StatusTypeDef I2C1_TransmitData(u
int8_t *pBuffer, uint16_t Length);
00181
00182 /* Link function for I2C HDMI_SINK periphera
ls */
00183 static void                I2C2_Init(void);
00184 static void                I2C2_Error(void);
00185 static void                I2C2_MspInit(I2C_Ha
ndleTypeDef *hi2c);
00186 /* Link function for HDMI_SINK peripheral */
00187 static HAL_StatusTypeDef I2C2_ReceiveData(ui
nt16_t Addr, uint8_t * pBuffer, uint16_t Length);
00188
00189 /**
00190   * @}
00191   */
00192
00193 /** @defgroup STM32072B_EVAL_LINK_Operations
_Functions LINK Operations Functions
```

```
00194   * @{
00195   */
00196
00197 /* Link functions for EEPROM peripheral */
00198 void                    EEPROM_IO_Init(void
);
00199 HAL_StatusTypeDef       EEPROM_IO_WriteData
(uint16_t DevAddress, uint16_t MemAddress, uint8_t
* pBuffer, uint32_t BufferSize);
00200 HAL_StatusTypeDef       EEPROM_IO_ReadData(
uint16_t DevAddress, uint16_t MemAddress, uint8_t*
 pBuffer, uint32_t BufferSize);
00201 HAL_StatusTypeDef       EEPROM_IO_IsDeviceR
eady(uint16_t DevAddress, uint32_t Trials);
00202
00203 /* Link functions for Temperature Sensor per
ipheral */
00204 void                    TSENSOR_IO_Init(void
);
00205 void                    TSENSOR_IO_Write(ui
nt16_t DevAddress, uint8_t* pBuffer, uint8_t Write
Addr, uint16_t Length);
00206 void                    TSENSOR_IO_Read(uin
t16_t DevAddress, uint8_t* pBuffer, uint8_t ReadAd
dr, uint16_t Length);
00207 uint16_t                TSENSOR_IO_IsDevice
Ready(uint16_t DevAddress, uint32_t Trials);
00208
00209 /* Link functions for CEC peripheral */
00210 void                    HDMI_CEC_IO_Init(vo
id);
00211 HAL_StatusTypeDef       HDMI_CEC_IO_WriteDa
ta(uint8_t * pBuffer, uint16_t BufferSize);
00212 HAL_StatusTypeDef       HDMI_CEC_IO_ReadData
(uint16_t DevAddress, uint8_t * pBuffer, uint16_t
BufferSize);
00213 #endif /* HAL_I2C_MODULE_ENABLED */
```

```
00214 /**
00215   * @}
00216   */
00217
00218 /** @addtogroup STM32072B_EVAL_BUS_Operation
s_Functions
00219   * @{
00220   */
00221
00222 #if defined(HAL_SPI_MODULE_ENABLED)
00223 /* SPIx bus function */
00224 static void              SPIx_Init(void);
00225 static void              SPIx_Write(uint8_t
Value);
00226 static void               SPIx_WriteReadData(
const uint8_t *DataIn, uint8_t *DataOut, uint16_t
DataLegnth);
00227 static void               SPIx_FlushFifo(void
);
00228 static uint32_t          SPIx_Read(void);
00229 static void              SPIx_Error (void);
00230 static void              SPIx_MspInit(SPI_Ha
ndleTypeDef *hspi);
00231 /**
00232   * @}
00233   */
00234
00235 /** @addtogroup STM32072B_EVAL_LINK_Operatio
ns_Functions
00236   * @{
00237   */
00238 /* Link functions for LCD peripheral */
00239 void                     LCD_IO_Init(void);
00240 void                     LCD_IO_WriteMultipl
eData(uint8_t *pData, uint32_t Size);
00241 void                     LCD_IO_WriteReg(uin
t8_t Reg);
```

```
00242 uint16_t                  LCD_IO_ReadData(uin
t16_t RegValue);
00243 void                      LCD_Delay(uint32_t
delay);
00244
00245 /* Link functions for SD Card peripheral */
00246 void                      SD_IO_Init(void);
00247 void                      SD_IO_CSState(uint8
_t state);
00248 void                      SD_IO_WriteReadData(
const uint8_t *DataIn, uint8_t *DataOut, uint16_t
DataLength);
00249 uint8_t                   SD_IO_WriteByte(uin
t8_t Data);
00250 #endif /* HAL_SPI_MODULE_ENABLED */
00251
00252 /**
00253   * @}
00254   */
00255
00256 /** @addtogroup STM32072B_EVAL_Exported_Func
tions
00257   * @{
00258   */
00259
00260 /**
00261   * @brief  This method returns the STM32F07
2B EVAL BSP Driver revision
00262   * @retval version : 0xXYZR (8bits for each
 decimal, R for RC)
00263   */
00264 uint32_t BSP_GetVersion(void)
00265 {
00266   return __STM32072B_EVAL_BSP_VERSION;
00267 }
00268
00269 /**
```

```
00270   * @brief  Configures LED GPIO.
00271   * @param  Led Specifies the Led to be conf
igured.
00272   *   This parameter can be one of following
 parameters:
00273   *     @arg LED1
00274   *     @arg LED2
00275   *     @arg LED3
00276   *     @arg LED4
00277   * @retval None
00278   */
00279 void BSP_LED_Init(Led_TypeDef Led)
00280 {
00281   GPIO_InitTypeDef  GPIO_InitStruct;
00282
00283   /* Enable the GPIO_LED clock */
00284   LEDx_GPIO_CLK_ENABLE(Led);
00285
00286   /* Configure the GPIO_LED pin */
00287   GPIO_InitStruct.Pin = LED_PIN[Led];
00288   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP
;
00289   GPIO_InitStruct.Pull = GPIO_PULLUP;
00290   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HI
GH;
00291
00292   HAL_GPIO_Init(LED_PORT[Led], &GPIO_InitStr
uct);
00293
00294   HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[L
ed], GPIO_PIN_SET);
00295 }
00296
00297 /**
00298   * @brief  Turns selected LED On.
00299   * @param  Led Specifies the Led to be set
on.
```

```
00300   *   This parameter can be one of following
 parameters:
00301   *     @arg LED1
00302   *     @arg LED2
00303   *     @arg LED3
00304   *     @arg LED4
00305   * @retval None
00306   */
00307 void BSP_LED_On(Led_TypeDef Led)
00308 {
00309   HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[L
ed], GPIO_PIN_RESET);
00310 }
00311
00312 /**
00313   * @brief  Turns selected LED Off.
00314   * @param  Led Specifies the Led to be set
off.
00315   *   This parameter can be one of following
 parameters:
00316   *     @arg LED1
00317   *     @arg LED2
00318   *     @arg LED3
00319   *     @arg LED4
00320   * @retval None
00321   */
00322 void BSP_LED_Off(Led_TypeDef Led)
00323 {
00324   HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[L
ed], GPIO_PIN_SET);
00325 }
00326
00327 /**
00328   * @brief  Toggles the selected LED.
00329   * @param  Led Specifies the Led to be togg
led.
00330   *   This parameter can be one of following
```

```
 parameters:
00331   *     @arg LED1
00332   *     @arg LED2
00333   *     @arg LED3
00334   *     @arg LED4
00335   * @retval None
00336   */
00337 void BSP_LED_Toggle(Led_TypeDef Led)
00338 {
00339   HAL_GPIO_TogglePin(LED_PORT[Led], LED_PIN[
Led]);
00340 }
00341
00342 /**
00343   * @brief  Configures Tamper Button GPIO or
 EXTI Line.
00344   * @param  Button Button to be configured
00345   *   This parameter can be one of the follo
wing values:
00346   *     @arg BUTTON_TAMPER: Tamper Push Butt
on
00347   * @param  Mode Button mode
00348   *   This parameter can be one of the follo
wing values:
00349   *     @arg BUTTON_MODE_GPIO: Button will b
e used as simple IO
00350   *     @arg BUTTON_MODE_EXTI: Button will b
e connected to EXTI line
00351   *                            with interrup
t generation capability
00352   * @retval None
00353   */
00354 void BSP_PB_Init(Button_TypeDef Button, Butt
onMode_TypeDef Mode)
00355 {
00356   GPIO_InitTypeDef GPIO_InitStruct;
00357
```

```
00358   /* Enable the Tamper Clock */
00359   TAMPERx_GPIO_CLK_ENABLE(Button);
00360
00361   GPIO_InitStruct.Pin = BUTTON_PIN[Button];
00362   GPIO_InitStruct.Pull = GPIO_PULLDOWN;
00363   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HI
GH;
00364
00365   if (Mode == BUTTON_MODE_GPIO)
00366   {
00367     /* Configure Button pin as input */
00368     GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
00369
00370     HAL_GPIO_Init(BUTTON_PORT[Button], &GPIO
_InitStruct);
00371   }
00372
00373   if (Mode == BUTTON_MODE_EXTI)
00374   {
00375     /* Configure Button pin as input with Ex
ternal interrupt */
00376     GPIO_InitStruct.Mode = GPIO_MODE_IT_FALL
ING;
00377     HAL_GPIO_Init(BUTTON_PORT[Button], &GPIO
_InitStruct);
00378
00379     /* Enable and set Button EXTI Interrupt
to the lowest priority */
00380     HAL_NVIC_SetPriority((IRQn_Type)(BUTTON_
IRQn[Button]), 0x03, 0x00);
00381     HAL_NVIC_EnableIRQ((IRQn_Type)(BUTTON_IR
Qn[Button]));
00382   }
00383 }
00384
00385 /**
00386   * @brief  Returns the selected button stat
```

```
e.
00387   * @param  Button Button to be checked.
00388   *   This parameter can be one of the follo
wing values:
00389   *     @arg BUTTON_TAMPER: Tamper Push Butt
on
00390   * @retval The Button GPIO pin value
00391   */
00392 uint32_t BSP_PB_GetState(Button_TypeDef Butt
on)
00393 {
00394   return HAL_GPIO_ReadPin(BUTTON_PORT[Button
], BUTTON_PIN[Button]);
00395 }
00396
00397 /**
00398   * @brief  Configures joystick GPIO and EXT
I modes.
00399   * @param  Joy_Mode Button mode.
00400   *    This parameter can be one of the foll
owing values:
00401   *      @arg  JOY_MODE_GPIO: Joystick pins
will be used as simple IOs
00402   *      @arg  JOY_MODE_EXTI: Joystick pins
will be connected to EXTI line
00403   *                                    with int
errupt generation capability
00404   * @retval HAL_OK: if all initializations a
re OK. Other value if error.
00405   */
00406 uint8_t BSP_JOY_Init(JOYMode_TypeDef Joy_Mod
e)
00407 {
00408   JOYState_TypeDef joykey;
00409   GPIO_InitTypeDef GPIO_InitStruct;
00410
00411   /* Initialized the Joystick. */
```

```
00412   for(joykey = JOY_SEL; joykey < (JOY_SEL+JO
Yn) ; joykey++)
00413   {
00414     /* Enable the JOY clock */
00415     JOYx_GPIO_CLK_ENABLE(joykey);
00416
00417     GPIO_InitStruct.Pin = JOY_PIN[joykey];
00418     GPIO_InitStruct.Pull = GPIO_PULLDOWN;
00419     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_
HIGH;
00420
00421     if (Joy_Mode == JOY_MODE_GPIO)
00422     {
00423       /* Configure Joy pin as input */
00424       GPIO_InitStruct.Mode = GPIO_MODE_INPUT
;
00425       HAL_GPIO_Init(JOY_PORT[joykey], &GPIO_
InitStruct);
00426     }
00427
00428     if (Joy_Mode == JOY_MODE_EXTI)
00429     {
00430       /* Configure Joy pin as input with Ext
ernal interrupt */
00431       GPIO_InitStruct.Mode = GPIO_MODE_IT_RI
SING;
00432       HAL_GPIO_Init(JOY_PORT[joykey], &GPIO_
InitStruct);
00433
00434       /* Enable and set Joy EXTI Interrupt t
o the lowest priority */
00435       HAL_NVIC_SetPriority((IRQn_Type)(JOY_I
RQn[joykey]), 0x03, 0x00);
00436       HAL_NVIC_EnableIRQ((IRQn_Type)(JOY_IRQn
[joykey]));
00437     }
00438   }
```

```
00439
00440   return HAL_OK;
00441 }
00442
00443 /**
00444   * @brief  Returns the current joystick sta
tus.
00445   * @retval Code of the joystick key pressed
00446   *          This code can be one of the fol
lowing values:
00447   *            @arg  JOY_NONE
00448   *            @arg  JOY_SEL
00449   *            @arg  JOY_DOWN
00450   *            @arg  JOY_LEFT
00451   *            @arg  JOY_RIGHT
00452   *            @arg  JOY_UP
00453   */
00454 JOYState_TypeDef BSP_JOY_GetState(void)
00455 {
00456   JOYState_TypeDef joykey;
00457
00458   for (joykey = JOY_SEL; joykey < (JOY_SEL +
JOYn) ; joykey++)
00459   {
00460     if (HAL_GPIO_ReadPin(JOY_PORT[joykey], J
OY_PIN[joykey]) == GPIO_PIN_SET)
00461     {
00462       /* Return Code Joystick key pressed */
00463       return joykey;
00464     }
00465   }
00466
00467   /* No Joystick key pressed */
00468   return JOY_NONE;
00469 }
00470
00471 #if defined(HAL_UART_MODULE_ENABLED)
```

```
00472 /**
00473   * @brief  Configures COM port.
00474   * @param  COM Specifies the COM port to be
 configured.
00475   *   This parameter can be one of following
 parameters:
00476   *     @arg COM1
00477   * @param  huart pointer to a UART_HandleTy
peDef structure that
00478   *   contains the configuration information
 for the specified UART peripheral.
00479   * @retval None
00480   */
00481 void BSP_COM_Init(COM_TypeDef COM, UART_Hand
leTypeDef* huart)
00482 {
00483   GPIO_InitTypeDef GPIO_InitStruct;
00484
00485   /* Enable GPIO clock */
00486   COMx_TX_GPIO_CLK_ENABLE(COM);
00487   COMx_RX_GPIO_CLK_ENABLE(COM);
00488
00489   /* Enable USART clock */
00490   COMx_CLK_ENABLE(COM);
00491
00492   /* Configure USART Tx as alternate functio
n push-pull */
00493   GPIO_InitStruct.Pin = COM_TX_PIN[COM];
00494   GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00495   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HI
GH;
00496   GPIO_InitStruct.Pull = GPIO_PULLUP;
00497   GPIO_InitStruct.Alternate = COM_TX_AF[COM]
;
00498   HAL_GPIO_Init(COM_TX_PORT[COM], &GPIO_Init
Struct);
00499
```

```
00500   /* Configure USART Rx as alternate functio
n push-pull */
00501   GPIO_InitStruct.Pin = COM_RX_PIN[COM];
00502   GPIO_InitStruct.Alternate = COM_RX_AF[COM]
;
00503   HAL_GPIO_Init(COM_RX_PORT[COM], &GPIO_Init
Struct);
00504
00505   /* USART configuration */
00506   huart->Instance = COM_USART[COM];
00507   HAL_UART_Init(huart);
00508 }
00509 #endif /* HAL_UART_MODULE_ENABLED */
00510
00511 /**
00512   * @}
00513   */
00514
00515 /** @addtogroup STM32072B_EVAL_BUS_Operation
s_Functions
00516   * @{
00517   */
00518
00519 /*********************************************
**********************************
00520                                   BUS OPERATIONS
00521 *********************************************
**********************************/
00522 #if defined(HAL_I2C_MODULE_ENABLED)
00523 /****************************** I2C Routine
s *********************************/
00524
00525 /**
00526   * @brief I2C Bus initialization
00527   * @retval None
00528   */
00529 static void I2C1_Init(void)
```

```
00530 {
00531   if(HAL_I2C_GetState(&heval_I2c1) == HAL_I2
C_STATE_RESET)
00532   {
00533     heval_I2c1.Instance              = EVAL_
I2C1;
00534     heval_I2c1.Init.Timing           = I2C1_
TIMING;
00535     heval_I2c1.Init.OwnAddress1      = 0;
00536     heval_I2c1.Init.AddressingMode   = I2C_A
DDRESSINGMODE_7BIT;
00537     heval_I2c1.Init.DualAddressMode  = I2C_D
UALADDRESS_DISABLE;
00538     heval_I2c1.Init.OwnAddress2      = 0;
00539     heval_I2c1.Init.OwnAddress2Masks = I2C_O
A2_NOMASK;
00540     heval_I2c1.Init.GeneralCallMode  = I2C_G
ENERALCALL_DISABLE;
00541     heval_I2c1.Init.NoStretchMode    = I2C_N
OSTRETCH_DISABLE;
00542
00543     /* Init the I2C */
00544     I2C1_MspInit(&heval_I2c1);
00545     HAL_I2C_Init(&heval_I2c1);
00546   }
00547 }
00548
00549 /**
00550   * @brief  Reads multiple data on the BUS.
00551   * @param  Addr   I2C Address
00552   * @param  Reg    Reg Address
00553   * @param  RegSize  The target register siz
e (can be 8BIT or 16BIT)
00554   * @param  pBuffer  pointer to read data bu
ffer
00555   * @param  Length  length of the data
00556   * @retval 0 if no problems to read multipl
```

```
e data
00557   */
00558 static HAL_StatusTypeDef I2C1_ReadBuffer(uin
t16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t
 *pBuffer, uint16_t Length)
00559 {
00560   HAL_StatusTypeDef status = HAL_OK;
00561
00562   status = HAL_I2C_Mem_Read(&heval_I2c1, Add
r, Reg, RegSize, pBuffer, Length, I2c1Timeout);
00563
00564   /* Check the communication status */
00565   if(status != HAL_OK)
00566   {
00567     /* Re-Initiaize the BUS */
00568     I2C1_Error();
00569   }
00570   return status;
00571 }
00572
00573 /**
00574   * @brief  Checks if target device is ready
 for communication.
00575   * @note   This function is used with Memor
y devices
00576   * @param  DevAddress Target device address
00577   * @param  Trials Number of trials
00578   * @retval HAL status
00579   */
00580 static HAL_StatusTypeDef I2C1_IsDeviceReady(
uint16_t DevAddress, uint32_t Trials)
00581 {
00582   return (HAL_I2C_IsDeviceReady(&heval_I2c1,
 DevAddress, Trials, I2c1Timeout));
00583 }
00584
00585 /**
```

```
00586   * @brief  Write a value in a register of t
he device through BUS.
00587   * @param  Addr Device address on BUS Bus.

00588   * @param  Reg The target register address
to write
00589   * @param  RegSize The target register size
 (can be 8BIT or 16BIT)
00590   * @param  pBuffer The target register valu
e to be written
00591   * @param  Length buffer size to be written
00592   * @retval None
00593   */
00594 static HAL_StatusTypeDef I2C1_WriteBuffer(ui
nt16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_
t *pBuffer, uint16_t Length)
00595 {
00596   HAL_StatusTypeDef status = HAL_OK;
00597
00598   status = HAL_I2C_Mem_Write(&heval_I2c1, Ad
dr, Reg, RegSize, pBuffer, Length, I2c1Timeout);
00599
00600   /* Check the communication status */
00601   if(status != HAL_OK)
00602   {
00603     /* Re-Initiaize the BUS */
00604     I2C1_Error();
00605   }
00606   return status;
00607 }
00608
00609 /**
00610   * @brief  Write buffer through I2C.
00611   * @param  pBuffer The address of the data
to be written
00612   * @param  Length buffer size to be written
00613   * @retval None
```

```
00614   */
00615 static HAL_StatusTypeDef I2C1_TransmitData(u
int8_t *pBuffer, uint16_t Length)
00616 {
00617   HAL_StatusTypeDef status = HAL_OK;
00618
00619   status = HAL_I2C_Slave_Transmit(&heval_I2c1
, pBuffer, Length, I2c1Timeout);
00620
00621   /* Check the communication status */
00622   if(status != HAL_OK)
00623   {
00624     /* Execute user timeout callback */
00625     I2C1_Error();
00626     return HAL_ERROR;
00627   }
00628   return HAL_OK;
00629 }
00630
00631 /**
00632   * @brief  Manages error callback by re-ini
tializing I2C.
00633   * @retval None
00634   */
00635 static void I2C1_Error(void)
00636 {
00637   /* De-initialize the I2C communication BUS
 */
00638   HAL_I2C_DeInit(&heval_I2c1);
00639
00640   /* Re-Initiaize the I2C communication BUS
*/
00641   I2C1_Init();
00642 }
00643
00644 /**
00645   * @brief I2C MSP Initialization
```

```
00646   * @param hi2c I2C handle
00647   * @retval None
00648   */
00649 static void I2C1_MspInit(I2C_HandleTypeDef *
hi2c)
00650 {
00651   GPIO_InitTypeDef GPIO_InitStruct;
00652   RCC_PeriphCLKInitTypeDef RCC_PeriphCLKInit
Struct;
00653
00654   /*##-1- Set source clock to SYSCLK for I2C
1 ###########################################
*/
00655   RCC_PeriphCLKInitStruct.PeriphClockSelecti
on = RCC_PERIPHCLK_I2C1;
00656   RCC_PeriphCLKInitStruct.I2c1ClockSelection
 = RCC_I2C1CLKSOURCE_SYSCLK;
00657   HAL_RCCEx_PeriphCLKConfig(&RCC_PeriphCLKIn
itStruct);
00658
00659   /*##-2- Configure the GPIOs ##############
#################################*/
00660
00661   /* Enable GPIO clock */
00662   EVAL_I2C1_GPIO_CLK_ENABLE();
00663
00664   /* Configure I2C SCL & SDA as alternate fu
nction  */
00665   GPIO_InitStruct.Pin       = (EVAL_I2C1_SCL
_PIN| EVAL_I2C1_SDA_PIN);
00666   GPIO_InitStruct.Mode      = GPIO_MODE_AF_O
D;
00667   GPIO_InitStruct.Pull      = GPIO_NOPULL;
00668   GPIO_InitStruct.Speed     = GPIO_SPEED_FRE
Q_HIGH;
00669   GPIO_InitStruct.Alternate = EVAL_I2C1_SCL_
SDA_AF;
```

```
00670   HAL_GPIO_Init(EVAL_I2C1_GPIO_PORT, &GPIO_I
nitStruct);
00671
00672   /*##-3- Configure the Eval I2C peripheral
######################################*/
00673   /* Enable I2C clock */
00674   EVAL_I2C1_CLK_ENABLE();
00675
00676   /* Force the I2C peripheral clock reset */
00677   EVAL_I2C1_FORCE_RESET();
00678
00679   /* Release the I2C peripheral clock reset
*/
00680   EVAL_I2C1_RELEASE_RESET();
00681 }
00682
00683 /**
00684   * @brief I2C Bus initialization
00685   * @retval None
00686   */
00687 static void I2C2_Init(void)
00688 {
00689   if(HAL_I2C_GetState(&heval_I2c2) == HAL_I2
C_STATE_RESET)
00690   {
00691     heval_I2c2.Instance              = EVAL_
I2C2;
00692     heval_I2c2.Init.Timing           = I2C2_
TIMING;
00693     heval_I2c2.Init.OwnAddress1      = 0;
00694     heval_I2c2.Init.AddressingMode   = I2C_A
DDRESSINGMODE_7BIT;
00695     heval_I2c2.Init.DualAddressMode  = I2C_D
UALADDRESS_DISABLE;
00696     heval_I2c2.Init.OwnAddress2      = 0;
00697     heval_I2c2.Init.OwnAddress2Masks = I2C_O
A2_NOMASK;
```

```
00698     heval_I2c2.Init.GeneralCallMode  = I2C_G
ENERALCALL_DISABLE;
00699     heval_I2c2.Init.NoStretchMode    = I2C_N
OSTRETCH_DISABLE;
00700
00701     /* Init the I2C */
00702     I2C2_MspInit(&heval_I2c2);
00703     HAL_I2C_Init(&heval_I2c2);
00704   }
00705 }
00706
00707 /**
00708   * @brief  Read a register of the device th
rough I2C.
00709   * @param  Addr Device address on I2C Bus.

00710   * @param  pBuffer The address to store the
 read data
00711   * @param  Length buffer size to be read
00712   * @retval None
00713   */
00714 static HAL_StatusTypeDef I2C2_ReceiveData(ui
nt16_t Addr, uint8_t * pBuffer, uint16_t Length)
00715 {
00716   HAL_StatusTypeDef status = HAL_OK;
00717
00718   status = HAL_I2C_Master_Receive(&heval_I2c2
, Addr, pBuffer, Length, I2c2Timeout);
00719
00720   /* Check the communication status */
00721   if(status != HAL_OK)
00722   {
00723     /* Execute user timeout callback */
00724     I2C2_Error();
00725   }
00726   return status;
00727 }
```

```
00728
00729 /**
00730   * @brief Discovery I2C2 error treatment fu
nction
00731   * @retval None
00732   */
00733 static void I2C2_Error(void)
00734 {
00735   /* De-initialize the I2C communication BUS
 */
00736   HAL_I2C_DeInit(&heval_I2c2);
00737
00738   /* Re-Initiaize the I2C communication BUS
*/
00739   I2C2_Init();
00740 }
00741
00742 /**
00743   * @brief I2C MSP Initialization
00744   * @param hi2c I2C handle
00745   * @retval None
00746   */
00747 static void I2C2_MspInit(I2C_HandleTypeDef *
hi2c)
00748 {
00749   GPIO_InitTypeDef GPIO_InitStruct;
00750   /* Enable GPIO clock */
00751   EVAL_I2C2_GPIO_CLK_ENABLE();
00752
00753   /* Configure I2C SCL and SDA as alternate
function  */
00754   GPIO_InitStruct.Pin = (EVAL_I2C2_SCL_PIN |
EVAL_I2C2_SDA_PIN);
00755   GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
00756   GPIO_InitStruct.Pull = GPIO_NOPULL;
00757   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HI
GH;
```

```
00758   GPIO_InitStruct.Alternate = EVAL_I2C2_AF;
00759   HAL_GPIO_Init(EVAL_I2C2_GPIO_PORT, &GPIO_I
nitStruct);
00760
00761   /* Enable I2C clock */
00762   EVAL_I2C2_CLK_ENABLE();
00763
00764   /* Force the I2C peripheral clock reset */
00765   EVAL_I2C2_FORCE_RESET();
00766
00767   /* Release the I2C peripheral clock reset
*/
00768   EVAL_I2C2_RELEASE_RESET();
00769 }
00770 #endif /*HAL_I2C_MODULE_ENABLED*/
00771
00772 #if defined(HAL_SPI_MODULE_ENABLED)
00773 /****************************** SPI Routine
s ********************************/
00774
00775 /**
00776   * @brief SPIx Bus initialization
00777   * @retval None
00778   */
00779 static void SPIx_Init(void)
00780 {
00781   if(HAL_SPI_GetState(&heval_Spi) == HAL_SPI
_STATE_RESET)
00782   {
00783     /* SPI Config */
00784     heval_Spi.Instance = EVAL_SPIx;
00785     /* SPI baudrate is set to 12 MHz (PCLK1/
SPI_BaudRatePrescaler = 48/4 = 12 MHz)
00786     to verify these constraints:
00787     HX8347D LCD SPI interface max baudrate i
s  50MHz for write and 6.66MHz for read
00788     PCLK1 frequency is set to 48 MHz
```

```
00789     - SD card SPI interface max baudrate is
25MHz for write/read
00790     */
00791     heval_Spi.Init.BaudRatePrescaler = SPI_B
AUDRATEPRESCALER_4;
00792     heval_Spi.Init.Direction = SPI_DIRECTION
_2LINES;
00793     heval_Spi.Init.CLKPhase = SPI_PHASE_2EDG
E;
00794     heval_Spi.Init.CLKPolarity = SPI_POLARIT
Y_HIGH;
00795     heval_Spi.Init.CRCCalculation = SPI_CRCC
ALCULATION_DISABLE;
00796     heval_Spi.Init.CRCPolynomial = 7;
00797     heval_Spi.Init.DataSize = SPI_DATASIZE_8
BIT;
00798     heval_Spi.Init.FirstBit = SPI_FIRSTBIT_M
SB;
00799     heval_Spi.Init.NSS = SPI_NSS_SOFT;
00800     heval_Spi.Init.TIMode = SPI_TIMODE_DISAB
LE;
00801     heval_Spi.Init.Mode = SPI_MODE_MASTER;
00802
00803     SPIx_MspInit(&heval_Spi);
00804     HAL_SPI_Init(&heval_Spi);
00805   }
00806 }
00807
00808 /**
00809   * @brief SPI Read 4 bytes from device
00810   * @retval Read data
00811   */
00812 static uint32_t SPIx_Read(void)
00813 {
00814   HAL_StatusTypeDef status = HAL_OK;
00815   uint32_t readvalue = 0x0;
00816   uint32_t writevalue = 0xFFFFFFFF;
```

```
00817
00818   status = HAL_SPI_TransmitReceive(&heval_Spi
, (uint8_t*) &writevalue, (uint8_t*) &readvalue, 1
, SpixTimeout);
00819
00820   /* Check the communication status */
00821   if(status != HAL_OK)
00822   {
00823     /* Execute user timeout callback */
00824     SPIx_Error();
00825   }
00826
00827   return readvalue;
00828 }
00829
00830 /**
00831   * @brief SPI Write a byte to device
00832   * @param DataIn value to be written
00833   * @param DataOut read value
00834   * @param DataLegnth data length
00835   * @retval None
00836   */
00837 static void SPIx_WriteReadData(const uint8_t
 *DataIn, uint8_t *DataOut, uint16_t DataLegnth)
00838 {
00839   HAL_StatusTypeDef status = HAL_OK;
00840
00841   status = HAL_SPI_TransmitReceive(&heval_Spi
, (uint8_t*) DataIn, DataOut, DataLegnth, SpixTime
out);
00842
00843   /* Check the communication status */
00844   if(status != HAL_OK)
00845   {
00846     /* Execute user timeout callback */
00847     SPIx_Error();
00848   }
```

```
00849 }
00850
00851 /**
00852   * @brief SPI Write a byte to device
00853   * @param Value value to be written
00854   * @retval None
00855   */
00856 static void SPIx_Write(uint8_t Value)
00857 {
00858   HAL_StatusTypeDef status = HAL_OK;
00859   uint8_t data;
00860
00861   status = HAL_SPI_TransmitReceive(&heval_Spi
, (uint8_t*) &Value, &data, 1, SpixTimeout);
00862
00863   /* Check the communication status */
00864   if(status != HAL_OK)
00865   {
00866     /* Execute user timeout callback */
00867     SPIx_Error();
00868   }
00869 }
00870
00871 /**
00872   * @brief  SPIx_FlushFifo
00873   * @retval None
00874   */
00875 static void SPIx_FlushFifo(void)
00876 {
00877   HAL_SPIEx_FlushRxFifo(&heval_Spi);
00878 }
00879
00880 /**
00881   * @brief SPI error treatment function
00882   * @retval None
00883   */
00884 static void SPIx_Error (void)
```

```
00885 {
00886   /* De-initialize the SPI communication BUS
 */
00887   HAL_SPI_DeInit(&heval_Spi);
00888
00889   /* Re- Initiaize the SPI communication BUS
 */
00890   SPIx_Init();
00891 }
00892
00893 /**
00894   * @brief SPI MSP Init
00895   * @param hspi SPI handle
00896   * @retval None
00897   */
00898 static void SPIx_MspInit(SPI_HandleTypeDef *
hspi)
00899 {
00900   GPIO_InitTypeDef GPIO_InitStruct;
00901
00902   /* Enable SPI clock  */
00903   EVAL_SPIx_CLK_ENABLE();
00904
00905   /* enable EVAL_SPI gpio clocks */
00906   EVAL_SPIx_SCK_GPIO_CLK_ENABLE();
00907   EVAL_SPIx_MISO_GPIO_CLK_ENABLE();
00908   EVAL_SPIx_MOSI_GPIO_CLK_ENABLE();
00909   EVAL_SPIx_MOSI_DIR_GPIO_CLK_ENABLE();
00910
00911   /* configure SPI SCK */
00912   GPIO_InitStruct.Pin       = EVAL_SPIx_SCK_
PIN;
00913   GPIO_InitStruct.Mode      = GPIO_MODE_AF_P
P;
00914   GPIO_InitStruct.Pull      = GPIO_NOPULL;
00915   GPIO_InitStruct.Speed     = GPIO_SPEED_FRE
Q_HIGH;
```

```
00916   GPIO_InitStruct.Alternate = EVAL_SPIx_SCK_
AF;
00917   HAL_GPIO_Init(EVAL_SPIx_SCK_GPIO_PORT, &GP
IO_InitStruct);
00918
00919   /* configure SPI MOSI */
00920   GPIO_InitStruct.Pin       = EVAL_SPIx_MOSI
_PIN;
00921   GPIO_InitStruct.Alternate = EVAL_SPIx_MOSI
_AF;
00922   HAL_GPIO_Init(EVAL_SPIx_MOSI_GPIO_PORT, &G
PIO_InitStruct);
00923
00924   /* configure SPI MISO  */
00925   GPIO_InitStruct.Pin       = EVAL_SPIx_MISO
_PIN;
00926   GPIO_InitStruct.Alternate = EVAL_SPIx_MISO
_AF;
00927   HAL_GPIO_Init(EVAL_SPIx_MISO_GPIO_PORT, &G
PIO_InitStruct);
00928
00929   /* Set PB.2 as Out PP, as direction pin fo
r MOSI */
00930   GPIO_InitStruct.Pin       = EVAL_SPIx_MOSI
_DIR_PIN;
00931   GPIO_InitStruct.Speed     = GPIO_SPEED_FRE
Q_MEDIUM;
00932   GPIO_InitStruct.Mode      = GPIO_MODE_OUTP
UT_PP;
00933   GPIO_InitStruct.Pull      = GPIO_NOPULL;
00934   HAL_GPIO_Init(EVAL_SPIx_MOSI_DIR_GPIO_PORT
, &GPIO_InitStruct);
00935
00936   /* MOSI DIRECTION as output */
00937   HAL_GPIO_WritePin(EVAL_SPIx_MOSI_DIR_GPIO_
PORT, EVAL_SPIx_MOSI_DIR_PIN, GPIO_PIN_SET);
00938
```

```
00939   /* Force the SPI peripheral clock reset */
00940   EVAL_SPIx_FORCE_RESET();
00941
00942   /* Release the SPI peripheral clock reset
*/
00943   EVAL_SPIx_RELEASE_RESET();
00944 }
00945
00946 #endif /*HAL_SPI_MODULE_ENABLED*/
00947
00948 /**
00949   * @}
00950   */
00951
00952 /** @addtogroup STM32072B_EVAL_LINK_Operatio
ns_Functions
00953   * @{
00954   */
00955
00956 /*****************************************
**********************************
00957 LINK OPERATIONS
00958 *****************************************
**********************************/
00959
00960 #if defined(HAL_SPI_MODULE_ENABLED)
00961 /********************************* LINK LCD
*********************************/
00962
00963 /**
00964   * @brief  Configures the LCD_SPI interface.

00965   * @retval None
00966   */
00967 void LCD_IO_Init(void)
00968 {
00969   GPIO_InitTypeDef GPIO_InitStruct;
```

```
00970
00971   /* Configure the LCD Control pins --------
---------------------------------*/
00972   LCD_NCS_GPIO_CLK_ENABLE();
00973
00974   /* Configure NCS in Output Push-Pull mode
*/
00975   GPIO_InitStruct.Pin     = LCD_NCS_PIN;
00976   GPIO_InitStruct.Mode    = GPIO_MODE_OUTPUT
_OD;
00977   GPIO_InitStruct.Pull    = GPIO_NOPULL;
00978   GPIO_InitStruct.Speed   = GPIO_SPEED_FREQ_
LOW;
00979   HAL_GPIO_Init(LCD_NCS_GPIO_PORT, &GPIO_Ini
tStruct);
00980
00981   /* Set or Reset the control line */
00982   LCD_CS_LOW();
00983   LCD_CS_HIGH();
00984
00985   SPIx_Init();
00986 }
00987
00988 /**
00989   * @brief  Write register value.
00990   * @param  pData Pointer on the register va
lue
00991   * @param  Size Size of byte to transmit to
 the register
00992   * @retval None
00993   */
00994 void LCD_IO_WriteMultipleData(uint8_t *pData
, uint32_t Size)
00995 {
00996   uint32_t counter = 0;
00997
00998   /* Reset LCD control line(/CS) and Send da
```

```
ta */
00999   LCD_CS_LOW();
01000
01001   /* Send Start Byte */
01002   SPIx_Write(START_BYTE | LCD_WRITE_REG);
01003
01004   if (Size == 1)
01005   {
01006     /* Only 1 byte to be sent to LCD - gener
al interface can be used */
01007     /* Send Data */
01008     SPIx_Write(*pData);
01009   }
01010   else
01011   {
01012     for (counter = Size; counter != 0; count
er--)
01013     {
01014       while(((heval_Spi.Instance->SR) & SPI_
FLAG_TXE) != SPI_FLAG_TXE)
01015       {
01016       }
01017       /* Need to invert bytes for LCD*/
01018       *((__IO uint8_t*)&heval_Spi.Instance->
DR) = *(pData+1);
01019
01020       while(((heval_Spi.Instance->SR) & SPI_
FLAG_TXE) != SPI_FLAG_TXE)
01021       {
01022       }
01023       *((__IO uint8_t*)&heval_Spi.Instance->
DR) = *pData;
01024       counter--;
01025       pData += 2;
01026     }
01027
01028     /* Wait until the bus is ready before re
```

```
leasing Chip select */
01029     while(((heval_Spi.Instance->SR) & SPI_FL
AG_BSY) != RESET)
01030     {
01031     }
01032   }
01033
01034   /* Empty the Rx fifo */
01035   SPIx_FlushFifo();
01036
01037   /* Reset LCD control line(/CS) and Send da
ta */
01038   LCD_CS_HIGH();
01039 }
01040
01041 /**
01042 * @brief  Writes address on LCD register.
01043 * @param  Reg Register to be written
01044   * @retval None
01045   */
01046 void LCD_IO_WriteReg(uint8_t Reg)
01047 {
01048   /* Reset LCD control line(/CS) and Send co
mmand */
01049   LCD_CS_LOW();
01050
01051   /* Send Start Byte */
01052   SPIx_Write(START_BYTE | SET_INDEX);
01053
01054   /* Write 16-bit Reg Index (High Byte is 0)
 */
01055   SPIx_Write(0x00);
01056   SPIx_Write(Reg);
01057
01058   /* Deselect : Chip Select high */
01059   LCD_CS_HIGH();
01060 }
```

```
01061
01062 /**
01063 * @brief  Read data from LCD data register.
01064 * @param  Reg Regsiter to be read
01065 * @retval readvalue
01066   */
01067 uint16_t LCD_IO_ReadData(uint16_t Reg)
01068 {
01069   uint32_t readvalue = 0;
01070
01071   /* Send Reg value to Read */
01072   LCD_IO_WriteReg(Reg);
01073
01074   /* Reset LCD control line(/CS) and Send co
mmand */
01075   LCD_CS_LOW();
01076
01077   /* Send Start Byte */
01078   SPIx_Write(START_BYTE | LCD_READ_REG);
01079
01080   /* Read Upper Byte */
01081   SPIx_Write(0xFF);
01082   readvalue = SPIx_Read();
01083   readvalue = readvalue << 8;
01084   readvalue |= SPIx_Read();
01085
01086   HAL_Delay(10);
01087
01088   /* Deselect : Chip Select high */
01089   LCD_CS_HIGH();
01090   return readvalue;
01091 }
01092
01093 /**
01094   * @brief  Wait for loop in ms.
01095   * @param  Delay in ms.
01096   */
```

```
01097 void LCD_Delay (uint32_t Delay)
01098 {
01099   HAL_Delay(Delay);
01100 }
01101
01102 /*********************************** LINK SD Ca
rd ************************************/
01103
01104 /**
01105   * @brief  Initializes the SD Card and put
it into StandBy State (Ready for
01106   *         data transfer).
01107   * @retval None
01108   */
01109 void SD_IO_Init(void)
01110 {
01111   GPIO_InitTypeDef  GPIO_InitStruct;
01112   uint8_t counter;
01113
01114   /* SD_CS_GPIO and SD_DETECT_GPIO Periph cl
ock enable */
01115   SD_CS_GPIO_CLK_ENABLE();
01116   SD_DETECT_GPIO_CLK_ENABLE();
01117
01118   /* Configure SD_CS_PIN pin: SD Card CS pin
 */
01119   GPIO_InitStruct.Pin = SD_CS_PIN;
01120   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_OD
;
01121   GPIO_InitStruct.Pull = GPIO_PULLUP;
01122   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_ME
DIUM;
01123   HAL_GPIO_Init(SD_CS_GPIO_PORT, &GPIO_InitS
truct);
01124
01125   /* Configure SD_DETECT_PIN pin: SD Card de
tect pin */
```

```
01126   GPIO_InitStruct.Pin = SD_DETECT_PIN;
01127   GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING
_FALLING;
01128   GPIO_InitStruct.Pull = GPIO_PULLUP;
01129   HAL_GPIO_Init(SD_DETECT_GPIO_PORT, &GPIO_I
nitStruct);
01130
01131   /* Configure LCD_CS_PIN pin: LCD Card CS p
in */
01132   GPIO_InitStruct.Pin   = LCD_NCS_PIN;
01133   GPIO_InitStruct.Mode  = GPIO_MODE_OUTPUT_P
P;
01134   GPIO_InitStruct.Pull  = GPIO_NOPULL;
01135   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_ME
DIUM;
01136   HAL_GPIO_Init(LCD_NCS_GPIO_PORT, &GPIO_Ini
tStruct);
01137   LCD_CS_HIGH();
01138
01139   /* Enable and set SD EXTI Interrupt to the
 lowest priority */
01140   HAL_NVIC_SetPriority(SD_DETECT_EXTI_IRQn,
0x03, 0);
01141   HAL_NVIC_EnableIRQ(SD_DETECT_EXTI_IRQn);
01142
01143   /*------------Put SD in SPI mode----------
----*/
01144   /* SD SPI Config */
01145   SPIx_Init();
01146
01147   /* SD chip select high */
01148   SD_CS_HIGH();
01149
01150   /* Send dummy byte 0xFF, 10 times with CS
high */
01151   /* Rise CS and MOSI for 80 clocks cycles */
```

```
01152   for (counter = 0; counter <= 9; counter++)
01153   {
01154     /* Send dummy byte 0xFF */
01155     SD_IO_WriteByte(SD_DUMMY_BYTE);
01156   }
01157 }
01158
01159
01160 void SD_IO_CSState(uint8_t val)
01161 {
01162   if(val == 1)
01163   {
01164     SD_CS_HIGH();
01165   }
01166   else
01167   {
01168     SD_CS_LOW();
01169   }
01170 }
01171
01172 /**
01173   * @brief  Write a byte on the SD.
01174   * @param  DataIn byte to send.
01175   * @param  DataOut read byte.
01176   * @param  DataLength data length.
01177   * @retval None
01178   */
01179 void SD_IO_WriteReadData(const uint8_t *Data
In, uint8_t *DataOut, uint16_t DataLength)
01180   {
01181   /* Send the byte */
01182   SPIx_WriteReadData(DataIn, DataOut, DataLe
ngth);
01183 }
01184
01185 /**
01186   * @brief  Writes a byte on the SD.
```

```
01187   * @param  Data byte to send.
01188   * @retval None
01189   */
01190 uint8_t SD_IO_WriteByte(uint8_t Data)
01191 {
01192   uint8_t tmp;
01193
01194   /* Send the byte */
01195   SPIx_WriteReadData(&Data,&tmp,1);
01196   return tmp;
01197 }
01198
01199 #endif /* HAL_SPI_MODULE_ENABLED */
01200
01201 #if defined(HAL_I2C_MODULE_ENABLED)
01202 /****************************** LINK I2C
EEPROM *****************************/
01203 /**
01204   * @brief  Initializes peripherals used by
the I2C EEPROM driver.
01205   * @retval None
01206   */
01207 void EEPROM_IO_Init(void)
01208 {
01209   I2C1_Init();
01210 }
01211
01212 /**
01213   * @brief  Write data to I2C EEPROM driver
01214   * @param  DevAddress Target device address
01215   * @param  MemAddress Internal memory addre
ss
01216   * @param  pBuffer Pointer to data buffer
01217   * @param  BufferSize Amount of data to be
sent
01218   * @retval HAL status
01219   */
```

```
01220 HAL_StatusTypeDef EEPROM_IO_WriteData(uint16
_t DevAddress, uint16_t MemAddress, uint8_t* pBuff
er, uint32_t BufferSize)
01221 {
01222   return (I2C1_WriteBuffer(DevAddress, MemAd
dress, I2C_MEMADD_SIZE_16BIT, pBuffer, BufferSize)
);
01223 }
01224
01225 /**
01226   * @brief  Read data from I2C EEPROM driver
01227   * @param  DevAddress Target device address
01228   * @param  MemAddress Internal memory addre
ss
01229   * @param  pBuffer Pointer to data buffer
01230   * @param  BufferSize Amount of data to be
read
01231   * @retval HAL status
01232   */
01233 HAL_StatusTypeDef EEPROM_IO_ReadData(uint16_
t DevAddress, uint16_t MemAddress, uint8_t* pBuffe
r, uint32_t BufferSize)
01234 {
01235   return (I2C1_ReadBuffer(DevAddress, MemAdd
ress, I2C_MEMADD_SIZE_16BIT, pBuffer, BufferSize))
;
01236 }
01237
01238 /**
01239   * @brief  Checks if target device is ready
 for communication.
01240   * @note   This function is used with Memor
y devices
01241   * @param  DevAddress Target device address
01242   * @param  Trials Number of trials
01243   * @retval HAL status
01244   */
```

```
01245 HAL_StatusTypeDef EEPROM_IO_IsDeviceReady(ui
nt16_t DevAddress, uint32_t Trials)
01246 {
01247   return (I2C1_IsDeviceReady(DevAddress, Tri
als));
01248 }
01249
01250 /********************************** LINK I2C
TEMPERATURE SENSOR ****************************/
01251 /**
01252   * @brief  Initializes peripherals used by
the I2C Temperature Sensor driver.
01253   * @retval None
01254   */
01255 void TSENSOR_IO_Init(void)
01256 {
01257   I2C1_Init();
01258 }
01259
01260 /**
01261   * @brief  Writes one byte to the TSENSOR.
01262   * @param  DevAddress Target device address
01263   * @param  pBuffer Pointer to data buffer
01264   * @param  WriteAddr TSENSOR's internal add
ress to write to.
01265   * @param  Length Number of data to write
01266   * @retval None
01267   */
01268 void TSENSOR_IO_Write(uint16_t DevAddress, u
int8_t* pBuffer, uint8_t WriteAddr, uint16_t Lengt
h)
01269 {
01270   I2C1_WriteBuffer(DevAddress, WriteAddr, I2
C_MEMADD_SIZE_8BIT, pBuffer, Length);
01271 }
01272
01273 /**
```

```
01274   * @brief  Reads one byte from the TSENSOR.
01275   * @param  DevAddress Target device address
01276   * @param  pBuffer  pointer to the buffer t
hat receives the data read from the TSENSOR.
01277   * @param  ReadAddr  TSENSOR's internal add
ress to read from.
01278   * @param  Length Number of data to read
01279   * @retval None
01280   */
01281 void TSENSOR_IO_Read(uint16_t DevAddress, ui
nt8_t* pBuffer, uint8_t ReadAddr, uint16_t Length)
01282 {
01283   I2C1_ReadBuffer(DevAddress, ReadAddr, I2C_
MEMADD_SIZE_8BIT, pBuffer, Length);
01284 }
01285
01286 /**
01287   * @brief  Checks if Temperature Sensor is
ready for communication.
01288   * @param  DevAddress Target device address
01289   * @param  Trials Number of trials
01290   * @retval HAL status
01291   */
01292 uint16_t TSENSOR_IO_IsDeviceReady(uint16_t D
evAddress, uint32_t Trials)
01293 {
01294   return (I2C1_IsDeviceReady(DevAddress, Tri
als));
01295 }
01296
01297 /****************************** LINK HDMI CE
C ******************************/
01298 /**
01299   * @brief  Initializes CEC low level.
01300   * @retval None
01301   */
01302 void HDMI_CEC_IO_Init (void)
```

```
01303 {
01304   GPIO_InitTypeDef  GPIO_InitStruct;
01305
01306   /* Enable CEC clock */
01307   __HAL_RCC_CEC_CLK_ENABLE();
01308
01309   /* Enable CEC LINE GPIO clock */
01310   HDMI_CEC_LINE_CLK_ENABLE();
01311
01312   /* Configure CEC LINE GPIO as alternate fu
nction open drain */
01313   GPIO_InitStruct.Pin = HDMI_CEC_LINE_PIN;
01314   GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
01315   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HI
GH;
01316   GPIO_InitStruct.Pull  = GPIO_NOPULL;
01317   GPIO_InitStruct.Alternate = HDMI_CEC_LINE_
AF;
01318   HAL_GPIO_Init(HDMI_CEC_LINE_GPIO_PORT, &GP
IO_InitStruct);
01319
01320   /* CEC IRQ Channel configuration */
01321   HAL_NVIC_SetPriority((IRQn_Type)HDMI_CEC_I
RQn, 0x3, 0x0);
01322   HAL_NVIC_EnableIRQ((IRQn_Type)HDMI_CEC_IRQ
n);
01323
01324   /* Enable CEC HPD SINK GPIO clock */
01325   HDMI_CEC_HPD_SINK_CLK_ENABLE();
01326
01327   /* Configure CEC HPD SINK GPIO as output p
ush pull */
01328   GPIO_InitStruct.Pin = HDMI_CEC_HPD_SINK_PIN
;
01329   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP
;
01330   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LO
```

```
W;
01331   GPIO_InitStruct.Pull  = GPIO_PULLDOWN;
01332   HAL_GPIO_Init(HDMI_CEC_HPD_SINK_GPIO_PORT,
 &GPIO_InitStruct);
01333
01334   I2C1_Init();
01335
01336   /* Enable CEC HPD SOURCE GPIO clock */
01337   HDMI_CEC_HPD_SOURCE_CLK_ENABLE();
01338
01339   /* Configure CEC HPD SOURCE GPIO as output
 push pull */
01340   GPIO_InitStruct.Pin = HDMI_CEC_HPD_SOURCE_
PIN;
01341   GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
01342   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LO
W;
01343   GPIO_InitStruct.Pull  = GPIO_PULLDOWN;
01344   HAL_GPIO_Init(HDMI_CEC_HPD_SOURCE_GPIO_PORT
, &GPIO_InitStruct);
01345
01346   I2C2_Init();
01347 }
01348
01349 /**
01350   * @brief  Write data to I2C HDMI CEC driver

01351   * @param  pBuffer Pointer to data buffer
01352   * @param  BufferSize Amount of data to be
sent
01353   * @retval HAL status
01354   */
01355 HAL_StatusTypeDef HDMI_CEC_IO_WriteData(uint
8_t * pBuffer, uint16_t BufferSize)
01356 {
01357   return (I2C1_TransmitData(pBuffer, BufferS
ize));
```

```
01358 }
01359
01360 /**
01361   * @brief  Read data to I2C HDMI CEC driver
01362   * @param  DevAddress Target device address
01363   * @param  pBuffer Pointer to data buffer
01364   * @param  BufferSize Amount of data to be
sent
01365   * @retval HAL status
01366   */
01367 HAL_StatusTypeDef HDMI_CEC_IO_ReadData(uint1
6_t DevAddress, uint8_t * pBuffer, uint16_t Buffer
Size)
01368 {
01369   return (I2C2_ReceiveData(DevAddress, pBuff
er, BufferSize));
01370 }
01371
01372 #endif /* HAL_I2C_MODULE_ENABLED */
01373
01374 /**
01375   * @}
01376   */
01377
01378 /**
01379   * @}
01380   */
01381
01382 /**
01383   * @}
01384   */
01385
01386 /**
01387   * @}
01388   */
01389
01390 /**
```

```
01391   * @}
01392   */
01393
01394 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32072B_EVAL BSP User Manual

stm32072b_eval_tsensor.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32072b_eval_tsensor.h
00004   * @author  MCD Application Team
00005   * @brief   This file contains all the functions prototypes for the
00006   *          stm32072b_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 __STM32072B_EVAL_TSENSOR_H
00039 #define __STM32072B_EVAL_TSENSOR_H
00040
00041 #ifdef __cplusplus
00042  extern "C" {
00043 #endif
00044
00045 /* Includes ------------------------------
-------------------------------------*/
00046 #include "stm32072b_eval.h"
00047 #include "../Components/stlm75/stlm75.h"
00048
00049 /** @addtogroup BSP
00050   * @{
00051   */
00052
00053 /** @addtogroup STM32072B_EVAL
00054   * @{
00055   */
00056
00057 /** @defgroup STM32072B_EVAL_TSENSOR STM3207
2B_EVAL TSENSOR
00058   * @{
00059   */
00060
00061 /** @defgroup STM32072B_EVAL_TSENSOR_Exporte
d_Types Exported Types
00062   * @{
00063   */
00064
00065 /**
00066   * @brief  TSENSOR Status
```

```
00067   */
00068 typedef enum
00069 {
00070   TSENSOR_OK = 0,
00071   TSENSOR_ERROR
00072 }TSENSOR_Status_TypDef;
00073
00074 /**
00075   * @}
00076   */
00077
00078 /** @defgroup STM32072B_EVAL_TSENSOR_Exporte
d_Constants Exported Constants
00079   * @{
00080   */
00081 /* Temperature Sensor hardware I2C address */

00082 #define TSENSOR_I2C_ADDRESS_A01 0x90
00083 #define TSENSOR_I2C_ADDRESS_A02 0x92
00084
00085 /* Maximum number of trials use for STTS751_
IsReady function */
00086 #define TSENSOR_MAX_TRIALS      50
00087
00088 /**
00089   * @}
00090   */
00091
00092 /** @defgroup STM32072B_EVAL_TSENSOR_Exporte
d_Functions Exported Functions
00093   * @{
00094   */
00095 uint32_t BSP_TSENSOR_Init(void);
00096 uint8_t  BSP_TSENSOR_ReadStatus(void);
00097 uint16_t BSP_TSENSOR_ReadTemp(void);
00098
00099 /**
```

```
00100   * @}
00101   */
00102
00103 /**
00104   * @}
00105   */
00106
00107 /**
00108   * @}
00109   */
00110
00111 /**
00112   * @}
00113   */
00114
00115 #ifdef __cplusplus
00116 }
00117 #endif
00118
00119 #endif /* __STM32072B_EVAL_TSENSOR_H */
00120
00121 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32072B_EVAL BSP User Manual

stm32072b_eval_tsensor.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32072b_eval_tsensor.c
00004   * @author  MCD Application Team
00005   * @brief   This file provides a set of functions needed to manage the I2C STLM75
00006   *          temperature sensor mounted on STM32072B-EVAL board .
00007   *          It implements a high level communication layer for read and write
00008   *          from/to this sensor. The needed STM32F0xx hardware resources (I2C and
00009   *          GPIO) are defined in stm32072b_eval.h file, and the initialization is
00010   *          performed in TSENSOR_IO_Init() function declared in stm32072b_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
```

```
00014   *           TSENSOR_IO_Init() function.
00015   *
00016   *     +-----------------------------------
------------------------------+
00017   *     |                                Pin assignm
ent                           |
00018   *     +-----------------------------------
----+-----------+-------------+
00019   *     |  STM32F0xx I2C Pins
    |   STLM75  |   Pin       |
00020   *     +-----------------------------------
----+-----------+-------------+
00021   *     | STLM75_I2C_SDA_PIN/ SDA
    |   SDA     |    1        |
00022   *     | STLM75_I2C_SCL_PIN/ SCL
    |   SCL     |    2        |
00023   *     | STLM75_I2C_SMBUSALERT_PIN/ SMBUS A
LERT|   OS/INT  |    3        |
00024   *     | .
    |   GND     |    4  (0V)  |
00025   *     | .
   |   GND     |    5  (0V)  |
00026   *     | .
    |   GND     |    6  (0V)  |
00027   *     | .
    |   GND     |    7  (0V)  |
00028   *     | .
   |   VDD     |    8  (3.3V)|
00029   *     +-----------------------------------
----+-----------+-------------+
00030   ******************************************
**********************************
00031   * @attention
00032   *
00033   * <h2><center>&copy; COPYRIGHT(c) 2016 STM
icroelectronics</center></h2>
00034   *
```

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

```
00054   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00055   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00056   *
00057   ****************************************
**********************************
00058   */
00059
00060 /* Includes ------------------------------
-----------------------------------*/
00061 #include "stm32072b_eval_tsensor.h"
00062
00063 /** @addtogroup BSP
00064   * @{
00065   */
00066
00067 /** @addtogroup STM32072B_EVAL
00068   * @{
00069   */
00070
00071 /** @addtogroup STM32072B_EVAL_TSENSOR
00072   * @brief      This file includes the STLM7
5 Temperature Sensor driver of
00073   *             STM32072B-EVAL boards.
00074   * @{
00075   */
00076
00077 /** @defgroup STM32072B_EVAL_TSENSOR_Private
_Variables Private Variables
00078   * @{
00079   */
00080 static TSENSOR_DrvTypeDef  *tsensor_drv;
00081 __IO uint16_t  TSENSORAddress = 0;
00082
00083 /**
00084   * @}
```

```
00085   */
00086
00087 /** @addtogroup STM32072B_EVAL_TSENSOR_Expor
ted_Functions
00088   * @{
00089   */
00090
00091 /**
00092   * @brief  Initializes peripherals used by
the I2C Temperature Sensor driver.
00093   * @retval TSENSOR status
00094   */
00095 uint32_t BSP_TSENSOR_Init(void)
00096 {
00097   uint8_t ret = TSENSOR_ERROR;
00098   TSENSOR_InitTypeDef STLM75_InitStructure;
00099
00100   /* Temperature Sensor Initialization */
00101   if(Stlm75Drv.IsReady(TSENSOR_I2C_ADDRESS_A
01, TSENSOR_MAX_TRIALS) == HAL_OK)
00102   {
00103     /* Initialize the temperature sensor dri
ver structure */
00104     TSENSORAddress = TSENSOR_I2C_ADDRESS_A01
;
00105     tsensor_drv = &Stlm75Drv;
00106
00107     ret = TSENSOR_OK;
00108   }
00109   else
00110   {
00111     if(Stlm75Drv.IsReady(TSENSOR_I2C_ADDRESS
_A02, TSENSOR_MAX_TRIALS) == HAL_OK)
00112     {
00113       /* Initialize the temperature sensor d
river structure */
00114       TSENSORAddress = TSENSOR_I2C_ADDRESS_A
```

```
02;
00115       tsensor_drv = &Stlm75Drv;
00116
00117       ret = TSENSOR_OK;
00118     }
00119     else
00120     {
00121       ret = TSENSOR_ERROR;
00122     }
00123   }
00124
00125   if (ret == TSENSOR_OK)
00126   {
00127     /* Configure Temperature Sensor : Conver
sion 9 bits in continuous mode */
00128     /* Alert outside range Limit Temperature
 12 <-> 24c */
00129     STLM75_InitStructure.AlertMode
   = STLM75_INTERRUPT_MODE;
00130     STLM75_InitStructure.ConversionMode
   = STLM75_CONTINUOUS_MODE;
00131     STLM75_InitStructure.TemperatureLimitHig
h  = 24;
00132     STLM75_InitStructure.TemperatureLimitLow
   = 12;
00133
00134     /* TSENSOR Init */
00135     tsensor_drv->Init(TSENSORAddress, &STLM7
5_InitStructure);
00136
00137     ret = TSENSOR_OK;
00138   }
00139
00140   return ret;
00141 }
00142
00143 /**
```

```
00144   * @brief  Returns the Temperature Sensor status.
00145   * @retval The Temperature Sensor status.
00146   */
00147 uint8_t BSP_TSENSOR_ReadStatus(void)
00148 {
00149   return (tsensor_drv->ReadStatus(TSENSORAddress));
00150 }
00151
00152 /**
00153   * @brief  Read Temperature register of STLM75.
00154   * @retval STLM75 measured temperature value.
00155   */
00156 uint16_t BSP_TSENSOR_ReadTemp(void)
00157 {
00158   return tsensor_drv->ReadTemp(TSENSORAddress);
00159
00160 }
00161
00162 /**
00163   * @}
00164   */
00165
00166
00167 /**
00168   * @}
00169   */
00170
00171
00172 /**
00173   * @}
00174   */
00175
```

```
00176 /**
00177   * @}
00178   */
00179
00180 /**
00181   * @}
00182   */
00183
00184 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32072B_EVAL BSP User Manual

Modules

BSP

Modules

STM32072B_EVAL

Generated on Wed Jul 5 2017 08:56:11 for STM32072B_EVAL BSP
User Manual by doxygen 1.7.6.1

STM32072B_EVAL BSP User Manual

STM32072B_EVAL

BSP

Modules

STM32072B_EVAL Common

This file provides firmware functions to manage Leds, push-buttons, COM ports, SD card on SPI and temperature sensor (LM75) available on STM32072B-EVAL evaluation board from STMicroelectronics.

STM32072B_EVAL EEPROM

This file includes the I2C EEPROM driver of STM32072B-EVAL board.

STM32072B_EVAL LCD

STM32072B_EVAL SD

STM32072B_EVAL TSENSOR

This file includes the STLM75 Temperature Sensor driver of STM32072B-EVAL boards.

STM32072B_EVAL BSP User Manual

STM32072B_EVAL Common

This file provides firmware functions to manage Leds, push-buttons, COM ports, SD card on SPI and temperature sensor (LM75) available on STM32072B-EVAL evaluation board from STMicroelectronics.
More...

Modules

Detailed Description

This file provides firmware functions to manage Leds, push-buttons, COM ports, SD card on SPI and temperature sensor (LM75) available on STM32072B-EVAL evaluation board from STMicroelectronics.

STM32072B_EVAL BSP User Manual

Exported Constants

STM32072B_EVAL Common

Modules

STM32072B_EVAL LED

Define for STM32072B_EVAL board.

STM32072B_EVAL BUTTON

STM32072B_EVAL COM

STM32072B_EVAL COMPONENT

STM32072B_EVAL BUS

STM32072B_EVAL BSP User Manual

STM32072B_EVAL EEPROM

This file includes the I2C EEPROM driver of STM32072B-EVAL board.
More...

Modules

Detailed Description

This file includes the I2C EEPROM driver of STM32072B-EVAL board.

STM32072B_EVAL BSP User Manual

STM32072B_EVAL LCD

STM32072B_EVAL

Modules

STM32072B_EVAL BSP User Manual

STM32072B_EVAL TSENSOR

This file includes the STLM75 Temperature Sensor driver of STM32072B-EVAL boards. More...

Modules

Detailed Description

This file includes the STLM75 Temperature Sensor driver of STM32072B-EVAL boards.
