# STM32F769I Discovery Low Level Private Typedef

**STM32F769I_DISCOVERY LOW LEVEL**

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by [doxygen](https://www.doxygen.org) 1.7.6.1
LOW_LEVEL Private Macros

STM32F769I_DISCOVERY LOW LEVEL

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
# STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Structures</td>
<td>Data Structure Index</td>
<td>Data Fields</td>
<td></td>
</tr>
</tbody>
</table>

## AUDIOIN_TypeDef Struct Reference

STMD2F769I_DISCOVERY_AUDIO Private Types
# Data Fields

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint16_t *</td>
<td>pRecBuf</td>
</tr>
<tr>
<td>uint32_t</td>
<td>RecSize</td>
</tr>
</tbody>
</table>
Detailed Description

Definition at line 149 of file stm32f769i_discovery_audio.c.
Field Documentation

**uint16_t** AUDIOIN_TypeDef::pRecBuf

Definition at line 151 of file stm32f769i_discovery_audio.c.

Referenced by **BSP_AUDIO_IN_Record()**, **HAL_DFSDM_FilterRegConvCpltCallback()**, and **HAL_DFSDM_FilterRegConvHalfCpltCallback()**.

**uint32_t** AUDIOIN_TypeDef::RecSize

Definition at line 152 of file stm32f769i_discovery_audio.c.

Referenced by **BSP_AUDIO_IN_Record()**, **HAL_DFSDM_FilterRegConvCpltCallback()**, and **HAL_DFSDM_FilterRegConvHalfCpltCallback()**.

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

- **stm32f769i_discovery_audio.c**

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by **doxygen** 1.7.6.1
## STM32F769I_DISCOVERY_AUDIO Private Defines

**STM32F769I_DISCOVERY_AUDIO**

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by [doxygen](https://www.doxygen.nl) 1.7.6.1
STM32F769I-Discovery BSP User Manual

STM32F769I_DISCOVERY_AUDIO Private Function Prototypes

STM32F769I_DISCOVERY_AUDIO

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**STM32F769I_DISCOVERY_AUDIO Exported Types**

`STM32F769I_DISCOVERY AUDIO`

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by [doxygen](https://www.doxygen.org) 1.7.6.1
# LCD Private Function Prototypes

STM32F769I_DISCOVERY LCD

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by [doxygen](http://www.doxygen.org) 1.7.6.1
# LCD_DrawPropTypeDef Struct Reference

STM32F769I DISCOVERY LCD Exported Types

LCD Drawing main properties. More...

```c
#include <stm32f769i_discovery_lcd.h>
```
## Data Fields

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint32_t</td>
<td>TextColor</td>
</tr>
<tr>
<td>uint32_t</td>
<td>BackColor</td>
</tr>
<tr>
<td>sFONT *</td>
<td>pFont</td>
</tr>
</tbody>
</table>
Detailed Description

LCD Drawing main properties.

Definition at line 266 of file stm32f769i_discovery_lcd.h.
**Field Documentation**

### `uint32_t LCD_DrawPropTypeDef::BackColor`

Specifies the background color below the text

Definition at line 269 of file `stm32f769i_discovery_lcd.h`.

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

### `sFONT* LCD_DrawPropTypeDef::pFont`

Specifies the font used for the text

Definition at line 270 of file `stm32f769i_discovery_lcd.h`.

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

### `uint32_t LCD_DrawPropTypeDef::TextColor`

Specifies the color of text

Definition at line 268 of file `stm32f769i_discovery_lcd.h`.

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

The documentation for this struct was generated from the following file:
• stm32f769i_discovery_lcd.h

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery
BSP User Manual by doxygen 1.7.6.1
Point Struct Reference
STM32F769I DISCOVERY LCD Exported Types

LCD Drawing point (pixel) geometric definition. More...

#include <stm32f769i_discovery_lcd.h>
### Data Fields

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int16_t</td>
<td>X</td>
</tr>
<tr>
<td>int16_t</td>
<td>Y</td>
</tr>
</tbody>
</table>
Detailed Description

LCD Drawing point (pixel) geometric definition.

Definition at line 277 of file stm32f769i_discovery_lcd.h.
Field Documentation

**int16_t** `Point::X`

geometric X position of drawing

Definition at line **279** of file *stm32f769i_discovery_lcd.h*.

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

**int16_t** `Point::Y`

geometric Y position of drawing

Definition at line **280** of file *stm32f769i_discovery_lcd.h*.

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

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

- *stm32f769i_discovery_lcd.h*
QSPI_Info Struct Reference

STM32F769I_DISCOVERY_QSPI Exported Types

#include <stm32f769i_discovery_qspi.h>
## Data Fields

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint32_t</td>
<td>FlashSize</td>
</tr>
<tr>
<td>uint32_t</td>
<td>EraseSectorSize</td>
</tr>
<tr>
<td>uint32_t</td>
<td>EraseSectorsNumber</td>
</tr>
<tr>
<td>uint32_t</td>
<td>ProgPageSize</td>
</tr>
<tr>
<td>uint32_t</td>
<td>ProgPagesNumber</td>
</tr>
</tbody>
</table>
Detailed Description

Definition at line 124 of file stm32f769i_discovery_qspi.h.
<table>
<thead>
<tr>
<th>Field Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>uint32_t QSPI_Info::EraseSectorSize</strong></td>
</tr>
<tr>
<td>Size of sectors for the erase operation</td>
</tr>
<tr>
<td>Definition at line <strong>126</strong> of file <strong>stm32f769i_discovery_qspi.h</strong>.</td>
</tr>
<tr>
<td>Referenced by <strong>BSP_QSPI_GetInfo()</strong></td>
</tr>
<tr>
<td><strong>uint32_t QSPI_Info::EraseSectorsNumber</strong></td>
</tr>
<tr>
<td>Number of sectors for the erase operation</td>
</tr>
<tr>
<td>Definition at line <strong>127</strong> of file <strong>stm32f769i_discovery_qspi.h</strong>.</td>
</tr>
<tr>
<td>Referenced by <strong>BSP_QSPI_GetInfo()</strong></td>
</tr>
<tr>
<td><strong>uint32_t QSPI_Info::FlashSize</strong></td>
</tr>
<tr>
<td>Size of the flash</td>
</tr>
<tr>
<td>Definition at line <strong>125</strong> of file <strong>stm32f769i_discovery_qspi.h</strong>.</td>
</tr>
<tr>
<td>Referenced by <strong>BSP_QSPI_GetInfo()</strong></td>
</tr>
<tr>
<td><strong>uint32_t QSPI_Info::ProgPageSize</strong></td>
</tr>
<tr>
<td>Size of pages for the program operation</td>
</tr>
<tr>
<td>Definition at line <strong>128</strong> of file <strong>stm32f769i_discovery_qspi.h</strong>.</td>
</tr>
<tr>
<td>Referenced by <strong>BSP_QSPI_GetInfo()</strong></td>
</tr>
</tbody>
</table>
uint32_t QSPI_Info::ProgPagesNumber

Number of pages for the program operation

Definition at line 129 of file stm32f769i_discovery_qspi.h.

Referenced by BSP_QSPI_GetInfo().

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

- stm32f769i_discovery_qspi.h

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery
BSP User Manual by doxygen 1.7.6.1
# STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## STM32F769I Discovery Sd Private TypesDef

**STM32F769I_DISCOVERY SD**

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by [doxygen](https://en.wikipedia.org/wiki/Doxygen) 1.7.6.1
STM32F769I Discovery Sd Private Defines

STM32F769I_DISCOVERY SD

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
## STM32F769I Discovery Sd Private Prototypes

STM32F769I_DISCOVERY SD

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
STM32F769I-Discovery BSP User Manual

SDRAM Private Types Definitions

STM32F769I_DISCOVERY SDRAM

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
SDRAM Private Defines

STM32F769I_DISCOVERY SDRAM
SDRAM Private Macros

STM32F769I_DISCOVERY SDRAM

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery
BSP User Manual by doxygen 1.7.6.1
SDRAM Exported Macro

STM32F769I_DISCOVERY SDRAM

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery
BSP User Manual by doxygen 1.7.6.1
## TS Private Types Definitions

STM32F769I_DISCOVERY TS

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by [doxygen](http://www.stackoverflow.com) 1.7.6.1
# STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## TS Private Types Defines

STM32F769I_DISCOVERY TS

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by [doxygen](http://www.doxygen.org) 1.7.6.1
## TS Private Macros

*STM32F769I_DISCOVERY TS*

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by [doxygen](https://www.doxygen.org/) 1.7.6.1
TS Private Function Prototypes

STM32F769I_DISCOVERY TS

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
TS_StateTypeDef Define TS State structure. More...

#include <stm32f769i_discovery_ts.h>
## Data Fields

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td>touchDetected</td>
<td></td>
</tr>
<tr>
<td>uint16_t</td>
<td>touchX [TS_MAX_NB_TOUCH]</td>
<td></td>
</tr>
<tr>
<td>uint16_t</td>
<td>touchY [TS_MAX_NB_TOUCH]</td>
<td></td>
</tr>
</tbody>
</table>
Detailed Description

**TS_TypeDef** Define TS State structure.

Definition at line 92 of file `stm32f769i_discovery_ts.h`.
Field Documentation

**uint8_t TS_StateTypeDef::touchDetected**

Total number of active touches detected at last scan

Definition at line 94 of file *stm32f769i_discovery_ts.h*.

Referenced by *BSP_TS_GetState()*.

**uint16_t TS_StateTypeDef::touchX[TS_MAX_NB_TOUCH]**

Touch X[0], X[1] coordinates on 12 bits

Definition at line 95 of file *stm32f769i_discovery_ts.h*.

Referenced by *BSP_TS_GetState()*.

**uint16_t TS_StateTypeDef::touchY[TS_MAX_NB_TOUCH]**

Touch Y[0], Y[1] coordinates on 12 bits

Definition at line 96 of file *stm32f769i_discovery_ts.h*.

Referenced by *BSP_TS_GetState()*.

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

- *stm32f769i_discovery_ts.h*
EEPROM Private Types

STM32F769I_DISCOVERY EEPROM

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
EEPROM Private Defines

STM32F769I_DISCOVERY EEPROM

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
### EEPROM Private Function Prototypes

**STM32F769I_DISCOVERY EEPROM**

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by [doxygen](http://www.doxygen.org) 1.7.6.1
EEPROM Exported Types

STM32F769I_DISCOVERY EEPROM

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
EEPROM Exported Macros

STM32F769I_DISCOVERY EEPROM

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
Here is a list of all struct and union fields with links to the structures/unions they belong to:

- BackColor : `LCD_DrawPropTypeDef`
- EraseSectorSize : `QSPI_Info`
- EraseSectorsNumber : `QSPI_Info`
- FlashSize : `QSPI_Info`
- pFont : `LCD_DrawPropTypeDef`
- pRecBuf : `AUDIOIN_TypeDef`
- ProgPageSize : `QSPI_Info`
- ProgPagesNumber : `QSPI_Info`
- RecSize : `AUDIOIN_TypeDef`
- TextColor : `LCD_DrawPropTypeDef`
- touchDetected : `TS_StateTypeDef`
- touchX : `TS_StateTypeDef`
- touchY : `TS_StateTypeDef`
- X : `Point`
- Y : `Point`
• BackColor : \texttt{LCD\_DrawPropTypeDef}
• EraseSectorSize : \texttt{QSPI\_Info}
• EraseSectorsNumber : \texttt{QSPI\_Info}
• FlashSize : \texttt{QSPI\_Info}
• pFont : \texttt{LCD\_DrawPropTypeDef}
• pRecBuf : \texttt{AUDIOIN\_TypeDef}
• ProgPageSize : \texttt{QSPI\_Info}
• ProgPagesNumber : \texttt{QSPI\_Info}
• RecSize : \texttt{AUDIOIN\_TypeDef}
• TextColor : \texttt{LCD\_DrawPropTypeDef}
• touchDetected : \texttt{TS\_StateTypeDef}
• touchX : \texttt{TS\_StateTypeDef}
• touchY : \texttt{TS\_StateTypeDef}
• X : \texttt{Point}
• Y : \texttt{Point}
Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- __DMAx_CLK_DISABLE : `stm32f769i_discovery_sdram.h`
- __DMAx_CLK_ENABLE : `stm32f769i_discovery_sdram.h`
- __DMAx_TxRx_CLK_ENABLE : `stm32f769i_discovery_sd.h`
- __STM32F769I_DISCOVERY_BSP_VERSION : `stm32f769i_discovery.c`
- __STM32F769I_DISCOVERY_BSP_VERSION_MAIN : `stm32f769i_discovery.c`
- __STM32F769I_DISCOVERY_BSP_VERSION_RC : `stm32f769i_discovery.c`
- __STM32F769I_DISCOVERY_BSP_VERSION_SUB1 : `stm32f769i_discovery.c`
- __STM32F769I_DISCOVERY_BSP_VERSION_SUB2 : `stm32f769i_discovery.c`
Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- a -

- ABS : `stm32f769i_discovery_lcd.c`
- ActiveLayer : `stm32f769i_discovery_lcd.c`
- AppBuffHalf : `stm32f769i_discovery_audio.c`
- AppBuffTrigger : `stm32f769i_discovery_audio.c`
- AUDIO_DFSDMx_BUTTOM_LEFT_CHANNEL : `stm32f769i_discovery_audio.h`
- AUDIO_DFSDMx_BUTTOM_LEFT_FILTER : `stm32f769i_discovery_audio.h`
- AUDIO_DFSDMx_BUTTOM_RIGHT_CHANNEL : `stm32f769i_discovery_audio.h`
- AUDIO_DFSDMx_BUTTOM_RIGHT_FILTER : `stm32f769i_discovery_audio.h`
- AUDIO_DFSDMx_CKOUT_AF : `stm32f769i_discovery_audio.h`
- AUDIO_DFSDMx_CKOUT_DMIC_GPIO_CLK_ENABLE : `stm32f769i_discovery_audio.h`
- AUDIO_DFSDMx_CKOUT_DMIC_GPIO_PORT : `stm32f769i_discovery_audio.h`
- AUDIO_DFSDMx_CKOUT_PIN : `stm32f769i_discovery_audio.h`
- AUDIO_DFSDMx_CLK_ENABLE : `stm32f769i_discovery_audio.h`
stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_BUTTOM_LEFT_IRQ:
  stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_BUTTOM_LEFT_IRQHandler:
  stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_BUTTOM_LEFT_STREAM:
  stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_IRQ:
  stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_IRQHandler:
  stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_STREAM:
  stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_CHANNEL:
  stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_CLK_ENABLE:
  stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_MEM_DATA_SIZE:
  stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_PERIPH_DATA_SIZE:
  stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_TOP_LEFT_IRQ:
  stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_TOP_LEFT_IRQHandler:
  stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_TOP_LEFT_STREAM:
  stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_TOP_RIGHT_IRQ:
  stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_TOP_RIGHT_IRQHandler:
  stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_TOP_RIGHT_STREAM:
  stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMIC_DATIN1_PIN:
  stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMIC_DATIN5_PIN:
  stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMIC_DATIN_AF:
  `stm32f769i_discovery_audio.h`
- AUDIO_DFSDMx_DMIC_DATIN_GPIO_CLK_ENABLE:
  `stm32f769i_discovery_audio.h`
- AUDIO_DFSDMx_DMIC_DATIN_GPIO_PORT:
  `stm32f769i_discovery_audio.h`
- AUDIO_DFSDMx_TOP_LEFT_CHANNEL:
  `stm32f769i_discovery_audio.h`
- AUDIO_DFSDMx_TOP_LEFT_FILTER:
  `stm32f769i_discovery_audio.h`
- AUDIO_DFSDMx_TOP_RIGHT_CHANNEL:
  `stm32f769i_discovery_audio.h`
- AUDIO_DFSDMx_TOP_RIGHT_FILTER:
  `stm32f769i_discovery_audio.h`
- audio_drv:
  `stm32f769i_discovery_audio.c`
- AUDIO_ERROR:
  `stm32f769i_discovery_audio.h`
- AUDIO_I2C_ADDRESS:
  `stm32f769i_discovery.h`
- AUDIO_IN_INT_GPIO_ENABLE:
  `stm32f769i_discovery_audio.h`
- AUDIO_IN_INT_GPIO_PIN:
  `stm32f769i_discovery_audio.h`
- AUDIO_IN_INT_GPIO_PORT:
  `stm32f769i_discovery_audio.h`
- AUDIO_IN_INT_IRQ:
  `stm32f769i_discovery_audio.h`
- AUDIO_IN_IRQ_PREPRIO:
  `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx:
  `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_AF:
  `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_CLK_DISABLE:
  `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_CLK_ENABLE:
  `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_DMAx_CHANNEL:
  `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_DMAx_CLK_ENABLE:
  `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_DMAx_IRQ:
  `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_DMAx_MEM_DATA_SIZE:
  `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_DMAx_PERIPH_DATA_SIZE:
stm32f769i_discovery_audio.h

- AUDIO_IN_SAIx_DMAx_STREAM: stm32f769i_discovery_audio.h
- AUDIO_IN_SAIx_SD_ENABLE: stm32f769i_discovery_audio.h
- AUDIO_IN_SAIx_SD_GPIO_PORT: stm32f769i_discovery_audio.h
- AUDIO_IN_SAIx_SD_PIN: stm32f769i_discovery_audio.h
- AUDIO_IO_DeInit(): stm32f769i_discovery.c
- AUDIO_IO_Delay(): stm32f769i_discovery.c
- AUDIO_IO_Init(): stm32f769i_discovery.c
- AUDIO_IO_Read(): stm32f769i_discovery.c
- AUDIO_IO_Write(): stm32f769i_discovery.c
- AUDIO_OK: stm32f769i_discovery_audio.h
- AUDIO_OUT_IRQ_PREPRIO: stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx: stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_AF: stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_CLK_DISABLE: stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_CLK_ENABLE: stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_DMAx_CHANNEL: stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_DMAx_CLK_ENABLE: stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_DMAx_IRQ: stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_DMAx_IRQHandler: stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_DMAx_MEM_DATA_SIZE: stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_DMAx_PERIPH_DATA_SIZE: stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_DMAx_STREAM: stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_FS_PIN: stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_MCLK_ENABLE: stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_MCLK_GPIO_PORT:
stm32f769i_discovery_audio.h

- AUDIO_OUT_SAIx_MCLK_PIN : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_SCK_PIN : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_SD_FS_CLK_ENABLE : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_SD_FS_SCK_GPIO_PORT : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_SD_PIN : stm32f769i_discovery_audio.h
- AUDIO_TIMEOUT : stm32f769i_discovery_audio.h
- AUDIODATA_SIZE : stm32f769i_discovery_audio.h
- AudioIn_ChannelNumber : stm32f769i_discovery_audio.c
- AudioIn_Device : stm32f769i_discovery_audio.c
STM32F769I-Discovery BSP User Manual

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

- b -

- BSP_AUDIO_FREQUENCY_11K : [stm32f769i_discovery_audio.h](stm32f769i_discovery_audio.h)
- BSP_AUDIO_FREQUENCY_16K : [stm32f769i_discovery_audio.h](stm32f769i_discovery_audio.h)
- BSP_AUDIO_FREQUENCY_22K : [stm32f769i_discovery_audio.h](stm32f769i_discovery_audio.h)
- BSP_AUDIO_FREQUENCY_32K : [stm32f769i_discovery_audio.h](stm32f769i_discovery_audio.h)
- BSP_AUDIO_FREQUENCY_44K : [stm32f769i_discovery_audio.h](stm32f769i_discovery_audio.h)
- BSP_AUDIO_FREQUENCY_48K : [stm32f769i_discovery_audio.h](stm32f769i_discovery_audio.h)
- BSP_AUDIO_FREQUENCY_8K : [stm32f769i_discovery_audio.h](stm32f769i_discovery_audio.h)
- BSP_AUDIO_FREQUENCY_96K : [stm32f769i_discovery_audio.h](stm32f769i_discovery_audio.h)
- BSP_AUDIO_IN_AllocScratch() : [stm32f769i_discovery_audio.c](stm32f769i_discovery_audio.c), [stm32f769i_discovery_audio.h](stm32f769i_discovery_audio.h)
- BSP_AUDIO_IN_ClockConfig() : [stm32f769i_discovery_audio.c](stm32f769i_discovery_audio.c), [stm32f769i_discovery_audio.h](stm32f769i_discovery_audio.h)
- BSP_AUDIO_IN_DeInit() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_IN_Error_CallBack() : stm32f769i_discovery_audio.h, stm32f769i_discovery_audio.c
- BSP_AUDIO_IN_GetChannelNumber() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_IN_HalfTransfer_CallBack() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_IN_Init() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_IN_InitEx() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_IN_MspDeInit() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_IN_MspInit() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_IN_Pause() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_IN_Record() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_IN_Resume() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_IN_Stop() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_IN_TransferComplete_CallBack() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_ChangeBuffer() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_ClockConfig() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_DeInit() : stm32f769i_discovery_audio.c,
stm32f769i_discovery_audio.h

- BSP_AUDIO_OUT_Error_CallBack() : 
  stm32f769i_discovery_audio.c , 
  stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_HalfTransfer_CallBack() : 
  stm32f769i_discovery_audio.c , 
  stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_Init() : stk32f769i_discovery_audio.c, 
  stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_MspDeInit() : stk32f769i_discovery_audio.c, 
  stk32f769i_discovery_audio.h
- BSP_AUDIO_OUT_MspInit() : stk32f769i_discovery_audio.c, 
  stk32f769i_discovery_audio.h
- BSP_AUDIO_OUT_Pause() : stk32f769i_discovery_audio.c, 
  stk32f769i_discovery_audio.h
- BSP_AUDIO_OUT_Play() : stk32f769i_discovery_audio.c, 
  stk32f769i_discovery_audio.h
- BSP_AUDIO_OUT_Resume() : stk32f769i_discovery_audio.c, 
  stk32f769i_discovery_audio.h
- BSP_AUDIO_OUT_SetAudioFrameSlot() : 
  stk32f769i_discovery_audio.c, 
  stk32f769i_discovery_audio.h
- BSP_AUDIO_OUT_SetFrequency() : 
  stk32f769i_discovery_audio.h , 
  stk32f769i_discovery_audio.c
- BSP_AUDIO_OUT_SetMute() : stk32f769i_discovery_audio.c, 
  stk32f769i_discovery_audio.h
- BSP_AUDIO_OUT_SetOutputMode() : 
  stk32f769i_discovery_audio.c, 
  stk32f769i_discovery_audio.h
- BSP_AUDIO_OUT_SetVolume() : 
  stk32f769i_discovery_audio.c, 
  stk32f769i_discovery_audio.h
- BSP_AUDIO_OUT_Stop() : stk32f769i_discovery_audio.c, 
  stk32f769i_discovery_audio.h
- BSP_AUDIO_OUT_TransferComplete_CallBack() : 
  stk32f769i_discovery_audio.c,
stm32f769i_discovery_audio.h

- BSP_EEPROM_DeInit() : stm32f769i_discovery_eeprom.c, stm32f769i_discovery_eeprom.h
- BSP_EEPROM_Init() : stm32f769i_discovery_eeprom.c, stm32f769i_discovery_eeprom.h
- BSP_EEPROM_ReadBuffer() : stm32f769i_discovery_eeprom.c, stm32f769i_discovery_eeprom.h
- BSP_EEPROM_TIMEOUT_UserCallback() : stm32f769i_discovery_eeprom.c, stm32f769i_discovery_eeprom.h
- BSP_EEPROM_WaitEepromStandbyState() : stm32f769i_discovery_eeprom.c, stm32f769i_discovery_eeprom.h
- BSP_EEPROM_WriteBuffer() : stm32f769i_discovery_eeprom.c, stm32f769i_discovery_eeprom.h
- BSP_EEPROM_WritePage() : stm32f769i_discovery_eeprom.c, stm32f769i_discovery_eeprom.h
- BSP_GetVersion() : stm32f769i_discovery.c, stm32f769i_discovery.h
- BSP_LCD_Clear() : stm32f769i_discovery_lcd.c
- BSP_LCD_ClearStringLine() : stm32f769i_discovery_lcd.c
- BSP_LCD_DisplayChar() : stm32f769i_discovery_lcd.c
- BSP_LCD_DisplayOff() : stm32f769i_discovery_lcd.c
- BSP_LCD_DisplayOn() : stm32f769i_discovery_lcd.c
- BSP_LCD_DisplayStringAt() : stm32f769i_discovery_lcd.c
- BSP_LCD_DisplayStringAtLine() : stm32f769i_discovery_lcd.c
- BSP_LCD_DMA2D_IRQHandler : stm32f769i_discovery_lcd.h
- BSP_LCD_DrawBitmap() : stm32f769i_discovery_lcd.c
- BSP_LCD_DrawCircle() : stm32f769i_discovery_lcd.c
- BSP_LCD_DrawEllipse() : stm32f769i_discovery_lcd.c
- BSP_LCD_DrawHLine() : stm32f769i_discovery_lcd.c
- BSP_LCD_DrawLine() : stm32f769i_discovery_lcd.c
- BSP_LCD_DrawPixel() : stm32f769i_discovery_lcd.c
- BSP_LCD_DrawPolygon() : stm32f769i_discovery_lcd.c
- BSP_LCD_DrawRect() : stm32f769i_discovery_lcd.c
- BSP_LCD_DrawVLine() : stm32f769i_discovery_lcd.c
- BSP_LCD_DSI_IRQHandler : stm32f769i_discovery_lcd.h
- BSP_LCD_DrawStringLine() : stm32f769i_discovery_lcd.c
- BSP_LCD_FillCircle() : stm32f769i_discovery_lcd.c
- BSP_LCD_FillEllipse() : stm32f769i_discovery_lcd.c
- BSP_LCD_FillPolygon() : stm32f769i_discovery_lcd.c
- BSP_LCD_FillRect() : stm32f769i_discovery_lcd.c
- BSP_LCD_GetBackColor() : stm32f769i_discovery_lcd.c
- BSP_LCD_GetFont() : stm32f769i_discovery_lcd.c
- BSP_LCD_GetTextColor() : stm32f769i_discovery_lcd.c
- BSP_LCD_GetXSize() : stm32f769i_discovery_lcd.c
- BSP_LCD_GetYSize() : stm32f769i_discovery_lcd.c
- BSP_LCD_HDMIInitEx() : stm32f769i_discovery_lcd.h
- BSP_LCD_Init() : stm32f769i_discovery_lcd.c
- BSP_LCD_InitEx() : stm32f769i_discovery_lcd.c
- BSP_LCD_LayerDefaultInit() : stm32f769i_discovery_lcd.c
- BSP_LCD_LTDC_ER_IRQHandler : stm32f769i_discovery_lcd.h
- BSP_LCD_LTDC_IRQHandler : stm32f769i_discovery_lcd.h
- BSP_LCD_MspDeInit() : stm32f769i_discovery_lcd.c
- BSP_LCD_MspInit() : stm32f769i_discovery_lcd.c
- BSP_LCD_ReadPixel() : stm32f769i_discovery_lcd.c
- BSP_LCD_Reset() : stm32f769i_discovery_lcd.c
- BSP_LCD_ResetColorKeying() : stm32f769i_discovery_lcd.c
- BSP_LCD_SelectLayer() : stm32f769i_discovery_lcd.c
- BSP_LCD_SetBackColor() : stm32f769i_discovery_lcd.c
- BSP_LCD_SetBrightness() : stm32f769i_discovery_lcd.c
- BSP_LCD_SetColorKeying() : stm32f769i_discovery_lcd.c
- BSP_LCD_SetFont() : stm32f769i_discovery_lcd.c
- BSP_LCD_SetLayerAddress() : stm32f769i_discovery_lcd.c
- BSP_LCD_SetLayerVisible() : stm32f769i_discovery_lcd.c
- BSP_LCD_SetLayerWindow() : stm32f769i_discovery_lcd.c
- BSP_LCD_SetTextColor() : stm32f769i_discovery_lcd.c
- BSP_LCD_SetTransparency() : stm32f769i_discovery_lcd.c
- BSP_LCD_SetXSize() : stm32f769i_discovery_lcd.c
- BSP_LCD_SetYSize() : stm32f769i_discovery_lcd.c
- BSP_LED_DeInit() : stm32f769i_discovery.c
- BSP_LED_Init() : stm32f769i_discovery.h
- BSP_LED_Off() : stm32f769i_discovery.c
stm32f769i_discovery.h

- BSP_LED_On() : stm32f769i_discovery.c, stm32f769i_discovery.h
- BSP_LED_Toggle() : stm32f769i_discovery.c, stm32f769i_discovery.h
- BSP_PB_DeInit() : stm32f769i_discovery.h, stm32f769i_discovery.c
- BSP_PB_GetState() : stm32f769i_discovery.c, stm32f769i_discovery.h
- BSP_PB_Init() : stm32f769i_discovery.c, stm32f769i_discovery.h
- BSP_QSPI_DeInit() : stm32f769i_discovery_qspi.c
- BSP_QSPI_EnableMemoryMappedMode() : stm32f769i_discovery_qspi.c
- BSP_QSPI_Erase_Blk() : stm32f769i_discovery_qspi.c
- BSP_QSPI_Erase_Chip() : stm32f769i_discovery_qspi.c
- BSP_QSPI_GetInfo() : stm32f769i_discovery_qspi.c
- BSP_QSPI_GetStatus() : stm32f769i_discovery_qspi.c
- BSP_QSPI_Init() : stm32f769i_discovery_qspi.c
- BSP_QSPI_MspDeInit() : stm32f769i_discovery_qspi.c
- BSP_QSPI_MspInit() : stm32f769i_discovery_qspi.c
- BSP_QSPI_Read() : stm32f769i_discovery_qspi.c
- BSP_QSPI_Write() : stm32f769i_discovery_qspi.c
- BSP_SD_AbortCallback() : stm32f769i_discovery_sd.c, stm32f769i_discovery_sd.h
- BSP_SD_CardInfo() : stm32f769i_discovery_sd.h
- BSP_SD_DeInit() : stm32f769i_discovery_sd.h, stm32f769i_discovery_sd.c
- BSP_SD_Detect_MspInit() : stm32f769i_discovery_sd.h, stm32f769i_discovery_sd.c
- BSP_SD_Erase() : stm32f769i_discovery_sd.h, stm32f769i_discovery_sd.c
- BSP_SD_GetCardInfo() : stm32f769i_discovery_sd.h, stm32f769i_discovery_sd.c
- BSP_SD_GetCardState() : stm32f769i_discovery_sd.c,
stm32f769i_discovery_sd.h

- BSP_SD_Init() : stm32f769i_discovery_sd.h, stm32f769i_discovery_sd.c
- BSP_SD_IsDetected() : stm32f769i_discovery_sd.h, stm32f769i_discovery_sd.c
- BSP_SD_ITConfig() : stm32f769i_discovery_sd.c, stm32f769i_discovery_sd.h
- BSP_SD_MspDeInit() : stm32f769i_discovery_sd.c, stm32f769i_discovery_sd.h
- BSP_SD_MspInit() : stm32f769i_discovery_sd.h, stm32f769i_discovery_sd.c
- BSP_SD_ReadBlocks() : stm32f769i_discovery_sd.c, stm32f769i_discovery_sd.h
- BSP_SD_ReadBlocks_DMA() : stm32f769i_discovery_sd.h, stm32f769i_discovery_sd.c
- BSP_SD_ReadCpltCallback() : stm32f769i_discovery_sd.c, stm32f769i_discovery_sd.h
- BSP_SD_WriteBlocks() : stm32f769i_discovery_sd.h, stm32f769i_discovery_sd.c
- BSP_SD_WriteBlocks_DMA() : stm32f769i_discovery_sd.h, stm32f769i_discovery_sd.c
- BSP_SD_WriteCpltCallback() : stm32f769i_discovery_sd.h, stm32f769i_discovery_sd.c
- BSP_SDMMC_DMA_Rx_IRQHandler: stm32f769i_discovery_sd.h
- BSP_SDMMC_DMA_Tx_IRQHandler: stm32f769i_discovery_sd.h
- BSP_SDMMC_IRQHandler: stm32f769i_discovery_sd.h
- BSP_SDRAM_DeInit() : stm32f769i_discovery_sdrarm.h, stm32f769i_discovery_sdrarm.c
- BSP_SDRAM_DMA_IRQHandler: stm32f769i_discovery_sdrarm.h
- BSP_SDRAM_DMA_IRQHandler: stm32f769i_discovery_sdrarm.h
- BSP_SDRAM_Init() : stm32f769i_discovery_sdrarm.h, stm32f769i_discovery_sdrarm.c
- BSP_SDRAM_Initialization_sequence() : stm32f769i_discovery_sdrarm.c, stm32f769i_discovery_sdrarm.h
- BSP_SDRAM_MspDeInit() : stm32f769i_discovery_sdram.h, stm32f769i_discovery_sdram.c
- BSP_SDRAM_MspInit() : stm32f769i_discovery_sdram.h, stm32f769i_discovery_sdram.c
- BSP_SDRAM_ReadData() : stm32f769i_discovery_sdram.c, stm32f769i_discovery_sdram.h
- BSP_SDRAM_ReadData_DMA() : stm32f769i_discovery_sdram.h, stm32f769i_discovery_sdram.c
- BSP_SDRAM_Sendcmd() : stm32f769i_discovery_sdram.h, stm32f769i_discovery_sdram.c
- BSP_SDRAM_WriteData() : stm32f769i_discovery_sdram.c, stm32f769i_discovery_sdram.h
- BSP_SDRAM_WriteData_DMA() : stm32f769i_discovery_sdram.c, stm32f769i_discovery_sdram.h
- BSP_TS_GetState() : stm32f769i_discovery_ts.h, stm32f769i_discovery_ts.c
- BSP_TS_Init() : stm32f769i_discovery_ts.c, stm32f769i_discovery_ts.h
- BSP_TS_INT_MspInit() : stm32f769i_discovery_ts.h, stm32f769i_discovery_ts.c
- BSP_TS_ITConfig() : stm32f769i_discovery_ts.h, stm32f769i_discovery_ts.c
- BUTTON_GPIO_CLK_ENABLE : stm32f769i_discovery.h
- BUTTON_IRQn : stm32f769i_discovery.c
- BUTTON_MODE_EXTI : stm32f769i_discovery.h
- BUTTON_MODE_GPIO : stm32f769i_discovery.h
- BUTTON_PIN : stm32f769i_discovery.c
- BUTTON_PORT : stm32f769i_discovery.c
- Button_TypeDef : stm32f769i_discovery.h
- BUTTON_USER : stm32f769i_discovery.h
- BUTTON_WAKEUP : stm32f769i_discovery.h
- ButtonMode_TypeDef : stm32f769i_discovery.h
- BUTTONNn : stm32f769i_discovery.h
- ButtonValue_TypeDef : stm32f769i_discovery.h
Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- c -

- CENTER_MODE : [stm32f769i_discovery_lcd.h](#)
- CODEC_AUDIOFRAME_SLOT_0123 : [stm32f769i_discovery_audio.h](#)
- CODEC_AUDIOFRAME_SLOT_02 : [stm32f769i_discovery_audio.h](#)
- CODEC_AUDIOFRAME_SLOT_13 : [stm32f769i_discovery_audio.h](#)
- Command : [stm32f769i_discovery_sdram.c](#)
Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- **d** -

- `DEFAULT_AUDIO_IN_BIT_RESOLUTION` : [stm32f769i_discovery_audio.h](#)
- `DEFAULT_AUDIO_IN_CHANNEL_NBR` : [stm32f769i_discovery_audio.h](#)
- `DEFAULT_AUDIO_IN_FREQ` : [stm32f769i_discovery_audio.h](#)
- `DEFAULT_AUDIO_IN_VOLUME` : [stm32f769i_discovery_audio.h](#)
- `DFSDM_CLOCK_DIVIDER` : [stm32f769i_discovery_audio.c](#)
- `DFSDM_FILTER_ORDER` : [stm32f769i_discovery_audio.c](#)
- `DFSDM_OVER_SAMPLING` : [stm32f769i_discovery_audio.c](#)
- `DFSDM_RIGHT_BIT_SHIFT` : [stm32f769i_discovery_audio.c](#)
- `DFSDMx_ChannelMspDeInit()` : [stm32f769i_discovery_audio.c](#)
- `DFSDMx_ChannelMspInit()` : [stm32f769i_discovery_audio.c](#)
- `DFSDMx_DeInit()` : [stm32f769i_discovery_audio.c](#)
- `DFSDMx_FilterMspDeInit()` : [stm32f769i_discovery_audio.c](#)
- `DFSDMx_FilterMspInit()` : [stm32f769i_discovery_audio.c](#)
- `DFSDMx_Init()` : [stm32f769i_discovery_audio.c](#)
- `DISCO_ERROR` : [stm32f769i_discovery.h](#)
- `DISCO_OK` : [stm32f769i_discovery.h](#)
- DISCO_Status_TypeDef : stm32f769i_discovery.h
- DISCOVERY_AUDIO_I2Cx : stm32f769i_discovery.h
- DISCOVERY_AUDIO_I2Cx_CLK_ENABLE : stm32f769i_discovery.h
- DISCOVERY_AUDIO_I2Cx_ER_IRQn : stm32f769i_discovery.h
- DISCOVERY_AUDIO_I2Cx_EV_IRQn : stm32f769i_discovery.h
- DISCOVERY_AUDIO_I2Cx_FORCE_RESET : stm32f769i_discovery.h
- DISCOVERY_AUDIO_I2Cx_RELEASE_RESET : stm32f769i_discovery.h
- DISCOVERY_AUDIO_I2Cx_SCL_AF : stm32f769i_discovery.h
- DISCOVERY_AUDIO_I2Cx_SCL_GPIO_CLK_ENABLE : stm32f769i_discovery.h
- DISCOVERY_AUDIO_I2Cx_SCL_GPIO_PORT : stm32f769i_discovery.h
- DISCOVERY_AUDIO_I2Cx_SCL_PIN : stm32f769i_discovery.h
- DISCOVERY_AUDIO_I2Cx_SDA_AF : stm32f769i_discovery.h
- DISCOVERY_AUDIO_I2Cx_SDA_GPIO_CLK_ENABLE : stm32f769i_discovery.h
- DISCOVERY_AUDIO_I2Cx_SDA_GPIO_PORT : stm32f769i_discovery.h
- DISCOVERY_AUDIO_I2Cx_SDA_PIN : stm32f769i_discovery.h
- DISCOVERY_DMAx_CLK_ENABLE : stm32f769i_discovery.h
- DISCOVERY_EXT_I2Cx : stm32f769i_discovery.h
- DISCOVERY_EXT_I2Cx_CLK_ENABLE : stm32f769i_discovery.h
- DISCOVERY_EXT_I2Cx_ER_IRQn : stm32f769i_discovery.h
- DISCOVERY_EXT_I2Cx_EV_IRQn : stm32f769i_discovery.h
- DISCOVERY_EXT_I2Cx_FORCE_RESET : stm32f769i_discovery.h
- DISCOVERY_EXT_I2Cx_RELEASE_RESET : stm32f769i_discovery.h
- DISCOVERY_EXT_I2Cx_SCL_PIN : stm32f769i_discovery.h
- DISCOVERY_EXT_I2Cx_SCL_SDA_AF : stm32f769i_discovery.h
- DISCOVERY_EXT_I2Cx_SCL_SDA_GPIO_CLK_ENABLE : stm32f769i_discovery.h
- DISCOVERY_EXT_I2Cx_SCL_SDA_GPIO_PORT : stm32f769i_discovery.h
- DISCOVERY_EXT_I2Cx_SCL_PIN : stm32f769i_discovery.h
- DISCOVERY_EXT_I2Cx_SCL_SDA_GPIO_PORT : `stm32f769i_discovery.h`
- DISCOVERY_EXT_I2Cx_SDA_PIN : `stm32f769i_discovery.h`
- DMA_MAX : `stm32f769i_discovery_audio.h`
- DMA_MAX_SZE : `stm32f769i_discovery_audio.h`
- DmaButtomLeftRecCplt : `stm32f769i_discovery_audio.c`
- DmaButtomLeftRecHalfCplt : `stm32f769i_discovery_audio.c`
- DmaButtomRightRecCplt : `stm32f769i_discovery_audio.c`
- DmaButtomRightRecHalfCplt : `stm32f769i_discovery_audio.c`
- DmaTopLeftRecCplt : `stm32f769i_discovery_audio.c`
- DmaTopLeftRecHalfCplt : `stm32f769i_discovery_audio.c`
- DmaTopRightRecCplt : `stm32f769i_discovery_audio.c`
- DmaTopRightRecHalfCplt : `stm32f769i_discovery_audio.c`
- DrawChar() : `stm32f769i_discovery_lcd.c`
- DrawProp : `stm32f769i_discovery_lcd.c`
- DSI_IO_WriteCmd() : `stm32f769i_discovery_lcd.c`
Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- e -

- EEPROM_FAIL : [file]
- EEPROM_I2C_ADDRESS_A01 : [file]
- EEPROM_I2C_ADDRESS_A02 : [file]
- EEPROM_IO_Init() : [file], [file]
- EEPROM_IO_IsDeviceReady() : [file], [file]
- EEPROM_IO_ReadData() : [file], [file]
- EEPROM_IO_WriteData() : [file], [file]
- EEPROM_MAX_SIZE : [file]
- EEPROM_MAX_TRIALS : [file]
- EEPROM_OK : [file]
- EEPROM_PAGESIZE : [file]
- EEPROM_TIMEOUT : [file]
- EEPROMAddress : [file]
- EEPROMDataRead : [file]
- EEPROMDataWrite : [file]
Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- f -

- FillTriangle() : `stm32f769i_discovery_lcd.c`
Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- **g** -

  - GEST_ID_MOVE_DOWN : [stm32f769i_discovery_ts.h](#)
  - GEST_ID_MOVE_LEFT : [stm32f769i_discovery_ts.h](#)
  - GEST_ID_MOVE_RIGHT : [stm32f769i_discovery_ts.h](#)
  - GEST_ID_MOVE_UP : [stm32f769i_discovery_ts.h](#)
  - GEST_ID_NB_MAX : [stm32f769i_discovery_ts.h](#)
  - GEST_ID_NO_GESTURE : [stm32f769i_discovery_ts.h](#)
  - GEST_ID_ZOOM_IN : [stm32f769i_discovery_ts.h](#)
  - GEST_ID_ZOOM_OUT : [stm32f769i_discovery_ts.h](#)
  - GPIO_PIN : [stm32f769i_discovery.c](#)
  - GPIO_PORT : [stm32f769i_discovery.c](#)
Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- **h** -
  
  - HAL_DFSDM_FilterRegConvCpltCallback() : \texttt{stm32f769i\_discovery\_audio.c}
  - HAL_DFSDM_FilterRegConvHalfCpltCallback() : \texttt{stm32f769i\_discovery\_audio.c}
  - HAL_SAI_ErrorCallback() : \texttt{stm32f769i\_discovery\_audio.c}
  - HAL_SAI_RxCpltCallback() : \texttt{stm32f769i\_discovery\_audio.c}
  - HAL_SAI_RxHalfCpltCallback() : \texttt{stm32f769i\_discovery\_audio.c}
  - HAL_SAI_TxCpltCallback() : \texttt{stm32f769i\_discovery\_audio.c}
  - HAL_SAI_TxHalfCpltCallback() : \texttt{stm32f769i\_discovery\_audio.c}
  - HAL_SD_AbortCallback() : \texttt{stm32f769i\_discovery\_sd.c}
  - HAL_SD_RxCpltCallback() : \texttt{stm32f769i\_discovery\_sd.c}
  - HAL_SD_TxCpltCallback() : \texttt{stm32f769i\_discovery\_sd.c}
  - hAudio\_in\_sai : \texttt{stm32f769i\_discovery\_audio.c}
  - hAudio\_out\_sai : \texttt{stm32f769i\_discovery\_audio.c}
  - hAudioIn : \texttt{stm32f769i\_discovery\_audio.c}
  - hAudioInBottomLeftChannel : \texttt{stm32f769i\_discovery\_audio.c}
  - hAudioInBottomLeftFilter : \texttt{stm32f769i\_discovery\_audio.c}
  - hAudioInBottomRightChannel : \texttt{stm32f769i\_discovery\_audio.c}
  - hAudioInBottomRightFilter : \texttt{stm32f769i\_discovery\_audio.c}
- hAudioInTopLeftChannel: `stm32f769i_discovery_audio.c`
- hAudioInTopLeftFilter: `stm32f769i_discovery_audio.c`
- hAudioInTopRightChannel: `stm32f769i_discovery_audio.c`
- hAudioInTopRightFilter: `stm32f769i_discovery_audio.c`
- hdma2d_discovery: `stm32f769i_discovery_lcd.h`, `stm32f769i_discovery_lcd.c`
- hDmaButtomLeft: `stm32f769i_discovery_audio.c`
- hDmaButtomRight: `stm32f769i_discovery_audio.c`
- hDmaTopLeft: `stm32f769i_discovery_audio.c`
- hDmaTopRight: `stm32f769i_discovery_audio.c`
- HDMI_ADV7533_ID: `stm32f769i_discovery_lcd.h`
- HDMI_FORMAT_720_480: `stm32f769i_discovery_lcd.h`
- HDMI_FORMAT_720_576: `stm32f769i_discovery_lcd.h`
- HDMI_IO_Delay(): `stm32f769i_discovery.c`
- HDMI_IO_Init(): `stm32f769i_discovery.c`
- HDMI_IO_Read(): `stm32f769i_discovery.c`
- HDMI_IO_Write(): `stm32f769i_discovery.c`
- hdsi_discovery: `stm32f769i_discovery_lcd.c`
- hdsivideo_handle: `stm32f769i_discovery_lcd.c`
- hI2cAudioHandler: `stm32f769i_discovery.c`
- hI2cExtHandler: `stm32f769i_discovery.c`
- hltdc_discovery: `stm32f769i_discovery_lcd.c`

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery BSP User Manual by `doxygen` 1.7.6.1
Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- i -

- I2C_Address : [stm32f769i_discovery_ts.c](#)
- I2Cx_Error() : [stm32f769i_discovery.c](#)
- I2Cx_Init() : [stm32f769i_discovery.c](#)
- I2Cx_IsDeviceReady() : [stm32f769i_discovery.c](#)
- I2Cx_MspInit() : [stm32f769i_discovery.c](#)
- I2Cx_ReadMultiple() : [stm32f769i_discovery.c](#)
- I2Cx_WriteMultiple() : [stm32f769i_discovery.c](#)
- INPUT_DEVICE_ANALOG_MIC : [stm32f769i_discovery_audio.h](#)
- INPUT_DEVICE_DIGITAL_MIC : [stm32f769i_discovery_audio.h](#)
Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- LCD_COLOR_BLACK : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_BLUE : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_BROWN : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_CYAN : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_DARKBLUE : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_DARKCYAN : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_DARKGRAY : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_DARKGREEN : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_DARKMAGENTA : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_DARKRED : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_DARKYELLOW : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_GRAY : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_GREEN : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_LIGHTBLUE : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_LIGHTCYAN : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_LIGHTGRAY : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_LIGHTGREEN : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_LIGHTMAGENTA : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_LIGHTRED : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_LIGHTYELLOW : [stm32f769i_discovery_lcd.h](#)
- LCD_COLOR_LIGHTYELLOW: stm32f769i_discovery_lcd.h
- LCD_COLOR_MAGENTA: stm32f769i_discovery_lcd.h
- LCD_COLOR_ORANGE: stm32f769i_discovery_lcd.h
- LCD_COLOR_RED: stm32f769i_discovery_lcd.h
- LCD_COLOR_TRANSPARENT: stm32f769i_discovery_lcd.h
- LCD_COLOR_WHITE: stm32f769i_discovery_lcd.h
- LCD_COLOR_YELLOW: stm32f769i_discovery_lcd.h
- LCD_DEFAULT_FONT: stm32f769i_discovery_lcd.h
- LCD_DSI_ADDRESS: stm32f769i_discovery_lcd.c
- LCD_DSI_ID: stm32f769i_discovery_lcd.c
- LCD_DSI_ID_REG: stm32f769i_discovery_lcd.c
- LCD_DSI_PIXEL_DATA_FMT_RBG565: stm32f769i_discovery_lcd.h
- LCD_DSI_PIXEL_DATA_FMT_RBG888: stm32f769i_discovery_lcd.h
- LCD_ERROR: stm32f769i_discovery_lcd.h
- LCD_FB_START_ADDRESS: stm32f769i_discovery_lcd.h
- LCD_IO_GetID(): stm32f769i_discovery_lcd.c
- LCD_LayerCfgTypeDef: stm32f769i_discovery_lcd.h
- LCD_OK: stm32f769i_discovery_lcd.h
- LCD_ORIENTATION_INVALID: stm32f769i_discovery_lcd.h
- LCD_ORIENTATION_LANDSCAPE: stm32f769i_discovery_lcd.h
- LCD_ORIENTATION_PORTRAIT: stm32f769i_discovery_lcd.h
- LCD_OrientationTypeDef: stm32f769i_discovery_lcd.h
- LCD_OTM8009A_ID: stm32f769i_discovery_lcd.h
- LCD_TIMEOUT: stm32f769i_discovery_lcd.h
- lcd_x_size: stm32f769i_discovery_lcd.c
- lcd_y_size: stm32f769i_discovery_lcd.c
- LED1: stm32f769i_discovery.h
- LED1_GPIO_PORT: stm32f769i_discovery.h
- LED1_PIN: stm32f769i_discovery.h
- LED2: stm32f769i_discovery.h
- LED2_GPIO_PORT: stm32f769i_discovery.h
- LED2_PIN: stm32f769i_discovery.h
- LED_GREEN: stm32f769i_discovery.h
- LED_RED: stm32f769i_discovery.h
- Led_TypeDef: `stm32f769i_discovery.h`
- LEDn: `stm32f769i_discovery.h`
- LEDx_GPIO_CLK_DISABLE: `stm32f769i_discovery.h`
- LEDx_GPIO_CLK_ENABLE: `stm32f769i_discovery.h`
- LEFT_MODE: `stm32f769i_discovery_lcd.h`
- LL_ConvertLineToARGB8888(): `stm32f769i_discovery_lcd.c`
- LL_FillBuffer(): `stm32f769i_discovery_lcd.c`
- LTDC_ACTIVE_LAYER_BACKGROUND: `stm32f769i_discovery_lcd.h`
- LTDC_ACTIVE_LAYER_FOREGROUND: `stm32f769i_discovery_lcd.h`
- LTDC_DEFAULT_ACTIVE_LAYER: `stm32f769i_discovery_lcd.h`
- LTDC_MAX_LAYER_NUMBER: `stm32f769i_discovery_lcd.h`
- LTDC_NB_OF_LAYERS: `stm32f769i_discovery_lcd.h`
Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- **m** -

  - MSD_ERROR : [stm32f769i_discovery_sd.h](#)
  - MSD_ERROR_SD_NOT_PRESENT : [stm32f769i_discovery_sd.h](#)
  - MSD_OK : [stm32f769i_discovery_sd.h](#)
STM32F769I-Discovery BSP User Manual

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

- o -

- OTG_HS_OVER_CURRENT_PIN : [stm32f769i_discovery.h](#)
- OTG_HS_OVER_CURRENT_PORT : [stm32f769i_discovery.h](#)
- OTG_HS_OVER_CURRENT_PORT_CLK_ENABLE : [stm32f769i_discovery.h](#)
- OTM8009A_IO_Delay() : [stm32f769i_discovery.c](#)
- OUTPUT_DEVICE_HEADPHONE1 : [stm32f769i_discovery_audio.h](#)
- OUTPUT_DEVICE_HEADPHONE2 : [stm32f769i_discovery_audio.h](#)
Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- p -

- PB_RESET : [stm32f769i_discovery.h](https://example.com/stm32f769i_discovery.h)
- PB_SET : [stm32f769i_discovery.h](https://example.com/stm32f769i_discovery.h)
- POLY_X : [stm32f769i_discovery_lcd.c](https://example.com/stm32f769i_discovery_lcd.c)
- POLY_Y : [stm32f769i_discovery_lcd.c](https://example.com/stm32f769i_discovery_lcd.c)
- pPoint : [stm32f769i_discovery_lcd.h](https://example.com/stm32f769i_discovery_lcd.h)
- pScratchBuff : [stm32f769i_discovery_audio.c](https://example.com/stm32f769i_discovery_audio.c)
STM32F769I-Discovery BSP User Manual

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

- q -

- QSPI_AutoPollingMemReady() : [stm32f769i_discovery_qspi.c](#)
- QSPI_BUSY : [stm32f769i_discovery_qspi.h](#)
- QSPI_CLK_DISABLE : [stm32f769i_discovery_qspi.h](#)
- QSPI_CLK_ENABLE : [stm32f769i_discovery_qspi.h](#)
- QSPI_CLK_GPIO_CLK_ENABLE : [stm32f769i_discovery_qspi.h](#)
- QSPI_CLK_GPIO_PORT : [stm32f769i_discovery_qspi.h](#)
- QSPI_CLK_PIN : [stm32f769i_discovery_qspi.h](#)
- QSPI_CLK_PIN_AF : [stm32f769i_discovery_qspi.h](#)
- QSPI_CS_GPIO_CLK_ENABLE : [stm32f769i_discovery_qspi.h](#)
- QSPI_CS_GPIO_PORT : [stm32f769i_discovery_qspi.h](#)
- QSPI_CS_PIN : [stm32f769i_discovery_qspi.h](#)
- QSPI_CS_PIN_AF : [stm32f769i_discovery_qspi.h](#)
- QSPI_D0_GPIO_CLK_ENABLE : [stm32f769i_discovery_qspi.h](#)
- QSPI_D0_GPIO_PORT : [stm32f769i_discovery_qspi.h](#)
- QSPI_D0_PIN : [stm32f769i_discovery_qspi.h](#)
- QSPI_D0_PIN_AF : [stm32f769i_discovery_qspi.h](#)
- QSPI_D1_GPIO_CLK_ENABLE : [stm32f769i_discovery_qspi.h](#)
- QSPI_D1_GPIO_PORT : [stm32f769i_discovery_qspi.h](#)
- QSPI_D1_PIN : [stm32f769i_discovery_qspi.h](#)
- `QSPI_D1_PIN_AF`: `stm32f769i_discovery_qspi.h`
- `QSPI_D2_GPIO_CLK_ENABLE`: `stm32f769i_discovery_qspi.h`
- `QSPI_D2_GPIO_PORT`: `stm32f769i_discovery_qspi.h`
- `QSPI_D2_PIN`: `stm32f769i_discovery_qspi.h`
- `QSPI_D2_PIN_AF`: `stm32f769i_discovery_qspi.h`
- `QSPI_D3_GPIO_CLK_ENABLE`: `stm32f769i_discovery_qspi.h`
- `QSPI_D3_GPIO_PORT`: `stm32f769i_discovery_qspi.h`
- `QSPI_D3_PIN`: `stm32f769i_discovery_qspi.h`
- `QSPI_D3_PIN_AF`: `stm32f769i_discovery_qspi.h`
- `QSPI_DummyCyclesCfg()`: `stm32f769i_discovery_qspi.c`
- `QSPI_EnterFourBytesAddress()`: `stm32f769i_discovery_qspi.c`
- `QSPI_EnterMemory_QPI()`: `stm32f769i_discovery_qspi.c`
- `QSPI_ERROR`: `stm32f769i_discovery_qspi.h`
- `QSPI_ExitMemory_QPI()`: `stm32f769i_discovery_qspi.c`
- `QSPI_FORCE_RESET`: `stm32f769i_discovery_qspi.h`
- `QSPI_NOT_SUPPORTED`: `stm32f769i_discovery_qspi.h`
- `QSPI_OK`: `stm32f769i_discovery_qspi.h`
- `QSPI_OutDrvStrengthCfg()`: `stm32f769i_discovery_qspi.c`
- `QSPI_RELEASE_RESET`: `stm32f769i_discovery_qspi.h`
- `QSPI_ResetMemory()`: `stm32f769i_discovery_qspi.c`
- `QSPI_SUSPENDED`: `stm32f769i_discovery_qspi.h`
- `QSPI_WriteEnable()`: `stm32f769i_discovery_qspi.c`
- `QSPIHandle`: `stm32f769i_discovery_qspi.c`
Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- r -

- REFRESH_COUNT : [stm32f769i_discovery_sdram.h](https://example.com)
- RIGHT_MODE : [stm32f769i_discovery_lcd.h](https://example.com)
Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- s -

- SAI_AUDIO_IN_MspDeInit() : stm32f769i_discovery_audio.c
- SAI_AUDIO_IN_MspInit() : stm32f769i_discovery_audio.c
- SAIx_In_DeInit() : stm32f769i_discovery_audio.c
- SAIx_In_Init() : stm32f769i_discovery_audio.c
- SAIx_Out_DeInit() : stm32f769i_discovery_audio.c
- SAIx_Out_Init() : stm32f769i_discovery_audio.c
- SaturaLH : stm32f769i_discovery_audio.c
- ScratchSize : stm32f769i_discovery_audio.c
- SD_DATATIMEOUT : stm32f769i_discovery_sd.h
- SD_DETECT_EXTI_IRQn : stm32f769i_discovery.h
- SD_DETECT_GPIO_CLK_DISABLE : stm32f769i_discovery.h
- SD_DETECT_GPIO_CLK_ENABLE : stm32f769i_discovery.h
- SD_DETECT_GPIO_PORT : stm32f769i_discovery.h
- SD_DETECT_PIN : stm32f769i_discovery.h
- SD_DetectIRQHandler : stm32f769i_discovery_sd.h
- SD_DMAx_Rx_CHANNEL : stm32f769i_discovery_sd.h
- SD_DMAx_Rx_IRQn : stm32f769i_discovery_sd.h
- SD_DMAx_Rx_STREAM : stm32f769i_discovery_sd.h
- SD_DMAx_Tx_CHANNEL : stm32f769i_discovery_sd.h
- SD_DMAx_Tx_IRQn : stm32f769i_discovery_sd.h
- SD_DMAx_Tx_STREAM : stm32f769i_discovery_sd.h
- SD_NOT_PRESENT : stm32f769i_discovery_sd.h
- SD_PRESENT : stm32f769i_discovery_sd.h
- SD_TRANSFER_BUSY : stm32f769i_discovery_sd.h
- SD_TRANSFER_OK : stm32f769i_discovery_sd.h
- SDCLOCK_PERIOD : stm32f769i_discovery_sdram.h
- SDRAM_DEVICE_ADDR : stm32f769i_discovery_sdram.h
- SDRAM_DEVICE_SIZE : stm32f769i_discovery_sdram.h
- SDRAM_DMAx_CHANNEL : stm32f769i_discovery_sdram.h
- SDRAM_DMAx_IRQn : stm32f769i_discovery_sdram.h
- SDRAM_DMAx_STREAM : stm32f769i_discovery_sdram.h
- SDRAM_ERROR : stm32f769i_discovery_sdram.h
- SDRAM_MEMORY_WIDTH : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_BURST_LENGTH_1 : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_BURST_LENGTH_2 : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_BURST_LENGTH_4 : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_BURST_LENGTH_8 : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_BURST_TYPE_INTERLEAVED : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_BURST_TYPESEQUENTIAL : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_CAS_LATENCY_2 : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_CAS_LATENCY_3 : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_OPERATING_MODE_STANDARD : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_WRITEBURST_MODE_PROGRAMMED : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_WRITEBURST_MODE_SINGLE : stm32f769i_discovery_sdram.h
- SDRAM_OK : stm32f769i_discovery_sdram.h
- SDRAM_TIMEOUT: `stm32f769i_discovery_sdram.h`
- sdramHandle: `stm32f769i_discovery_sdram.c`
Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- t -

- Text_AlignModeTypdef : [stm32f769i_discovery_lcd.h](#)
- Timing : [stm32f769i_discovery_sdram.c](#)
- TOUCH_EVENT_CONTACT : [stm32f769i_discovery_ts.h](#)
- TOUCH_EVENT_LIFT_UP : [stm32f769i_discovery_ts.h](#)
- TOUCH_EVENT_NB_MAX : [stm32f769i_discovery_ts.h](#)
- TOUCH_EVENT_NO_EVT : [stm32f769i_discovery_ts.h](#)
- TOUCH_EVENT_PRESS_DOWN : [stm32f769i_discovery_ts.h](#)
- TS_DEVICE_NOT_FOUND : [stm32f769i_discovery_ts.h](#)
- ts_driver : [stm32f769i_discovery_ts.c](#)
- TS_ERROR : [stm32f769i_discovery_ts.h](#)
- ts_event_string_tab : [stm32f769i_discovery_ts.c](#), [stm32f769i_discovery_ts.h](#)
- ts_gesture_id_string_tab : [stm32f769i_discovery_ts.c](#), [stm32f769i_discovery_ts.h](#)
- TS_GestureIdTypeDef : [stm32f769i_discovery_ts.h](#)
- TS_I2C_ADDRESS : [stm32f769i_discovery.h](#)
- TS_I2C_ADDRESS_A02 : [stm32f769i_discovery.h](#)
- TS_INT_EXTI_IRQn : [stm32f769i_discovery.h](#)
- TS_INT_GPIO_CLK_DISABLE : [stm32f769i_discovery.h](#)
- TS_INT_GPIO_CLK_ENABLE : stm32f769i_discovery.h
- TS_INT_GPIO_PORT : stm32f769i_discovery.h
- TS_INT_PIN : stm32f769i_discovery.h
- TS_IO_Delay() : stm32f769i_discovery.c
- TS_IO_Init() : stm32f769i_discovery.c
- TS_IO_Read() : stm32f769i_discovery.c
- TS_IO_ReadMultiple() : stm32f769i_discovery.c
- TS_IO_Write() : stm32f769i_discovery.c
- TS_IO_WriteMultiple() : stm32f769i_discovery.c
- TS_IRQ_PENDING : stm32f769i_discovery_ts.h
- TS_MAX_NB_TOUCH : stm32f769i_discovery_ts.h
- TS_NO_IRQ_PENDING : stm32f769i_discovery_ts.h
- TS_OK : stm32f769i_discovery_ts.h
- ts_orientation : stm32f769i_discovery_ts.c
- TS_StatusTypeDef : stm32f769i_discovery_ts.h
- TS_SWAP_NONE : stm32f769i_discovery_ts.h
- TS_SWAP_X : stm32f769i_discovery_ts.h
- TS_SWAP_XY : stm32f769i_discovery_ts.h
- TS_SWAP_Y : stm32f769i_discovery_ts.h
- TS_TIMEOUT : stm32f769i_discovery_ts.h
- TS_TouchEventTypeDef : stm32f769i_discovery_ts.h
Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- u -

- uSdHandle : [stm32f769i_discovery_sd.c](#)
- USER_BUTTON_EXTIIRQn : [stm32f769i_discovery.h](#)
- USER_BUTTON_GPIO_CLK_DISABLE : [stm32f769i_discovery.h](#)
- USER_BUTTON_GPIO_CLK_ENABLE : [stm32f769i_discovery.h](#)
- USER_BUTTON_GPIO_PORT : [stm32f769i_discovery.h](#)
- USER_BUTTON_PIN : [stm32f769i_discovery.h](#)
Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- **WAKEUP_BUTTON_EXTI_IRQn**: [stm32f769i_discovery.h](#)
- **WAKEUP_BUTTON_GPIO_CLK_DISABLE**: [stm32f769i_discovery.h](#)
- **WAKEUP_BUTTON_GPIO_CLK_ENABLE**: [stm32f769i_discovery.h](#)
- **WAKEUP_BUTTON_GPIO_PORT**: [stm32f769i_discovery.h](#)
- **WAKEUP_BUTTON_PIN**: [stm32f769i_discovery.h](#)
- a -

- AUDIO_IO_DeInit() : stm32f769i_discovery.c
- AUDIO_IO_Delay() : stm32f769i_discovery.c
- AUDIO_IO_Init() : stm32f769i_discovery.c
- AUDIO_IO_Read() : stm32f769i_discovery.c
- AUDIO_IO_Write() : stm32f769i_discovery.c
### STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>File List</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>Functions</td>
<td>Variables</td>
<td>Typedefs</td>
</tr>
<tr>
<td>Defines</td>
<td>a b d e f g h i l o q s t</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### - b -

- BSP_AUDIO_IN_AllocScratch() : *stm32f769i_discovery_audio.c*, *stm32f769i_discovery_audio.h*
- BSP_AUDIO_IN_ClockConfig() : *stm32f769i_discovery_audio.h*, *stm32f769i_discovery_audio.c*
- BSP_AUDIO_IN_DeInit() : *stm32f769i_discovery_audio.c*, *stm32f769i_discovery_audio.h*
- BSP_AUDIO_IN_Error_CallBack() : *stm32f769i_discovery_audio.h*, *stm32f769i_discovery_audio.c*
- BSP_AUDIO_IN_GetChannelNumber() : *stm32f769i_discovery_audio.c*, *stm32f769i_discovery_audio.h*
- BSP_AUDIO_IN_HalfTransfer_CallBack() : *stm32f769i_discovery_audio.c*, *stm32f769i_discovery_audio.h*
- BSP_AUDIO_IN_Init() : *stm32f769i_discovery_audio.c*, *stm32f769i_discovery_audio.h*
- BSP_AUDIO_IN_InitEx() : *stm32f769i_discovery_audio.h*, *stm32f769i_discovery_audio.c*
- BSP_AUDIO_IN_MspDeInit() : *stm32f769i_discovery_audio.c*, *stm32f769i_discovery_audio.h*
stm32f769i_discovery_audio.h

- BSP_AUDIO_IN_MspInit() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_IN_Pause() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_IN_Record() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_IN.Resume() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_IN_Stop() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_IN_TransferComplete_CallBack() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_ChangeBuffer() : stm32f769i_discovery_audio.h, stm32f769i_discovery_audio.c
- BSP_AUDIO_OUT_ClockConfig() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_DeInit() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_Error_CallBack() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_HalfTransfer_CallBack() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_Init() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_MspDeInit() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_MspInit() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_Pause() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_Play() : stm32f769i_discovery_audio.c,
stm32f769i_discovery_audio.h

- BSP_AUDIO_OUT_Resume() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_SetAudioFrameSlot() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_SetFrequency() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_SetMute() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_SetOutputMode() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_SetVolume() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_AUDIO_OUT_Stop() : stm32f769i_discovery_audio.h, stm32f769i_discovery_audio.c
- BSP_AUDIO_OUT_TransferComplete_CallBack() : stm32f769i_discovery_audio.c, stm32f769i_discovery_audio.h
- BSP_EEPROM_DeInit() : stm32f769i_discovery_eeprom.c, stm32f769i_discovery_eeprom.h
- BSP_EEPROM_Init() : stm32f769i_discovery_eeprom.c, stm32f769i_discovery_eeprom.h
- BSP_EEPROM_ReadBuffer() : stm32f769i_discovery_eeprom.c, stm32f769i_discovery_eeprom.h
- BSP_EEPROM_TIMEOUT_UserCallback() : stm32f769i_discovery_eeprom.c, stm32f769i_discovery_eeprom.h
- BSP_EEPROM_WriteBuffer() : stm32f769i_discovery_eeprom.c, stm32f769i_discovery_eeprom.h
- BSP_EEPROM_WritePage() : stm32f769i_discovery_eeprom.c, stm32f769i_discovery_eeprom.h

BSP_EEPROM_WaitEepromStandbyState() : stm32f769i_discovery_eeprom.c, stm32f769i_discovery_eeprom.h
stm32f769i_discovery_eeprom.h

- BSP_GetVersion() : stm32f769i_discovery.c, stm32f769i_discovery.h
- BSP_LCD_Clear() : stm32f769i_discovery_lcd.c
- BSP_LCD_ClearStringLine() : stm32f769i_discovery_lcd.c
- BSP_LCD_DisplayChar() : stm32f769i_discovery_lcd.c
- BSP_LCD_DisplayOff() : stm32f769i_discovery_lcd.c
- BSP_LCD_DisplayOn() : stm32f769i_discovery_lcd.c
- BSP_LCD_DisplayStringAt() : stm32f769i_discovery_lcd.c
- BSP_LCD_DisplayStringAtLine() : stm32f769i_discovery_lcd.c
- BSP_LCD_DrawBitmap() : stm32f769i_discovery_lcd.c
- BSP_LCD_DrawCircle() : stm32f769i_discovery_lcd.c
- BSP_LCD_DrawEllipse() : stm32f769i_discovery_lcd.c
- BSP_LCD_DrawHLine() : stm32f769i_discovery_lcd.c
- BSP_LCD_DrawLine() : stm32f769i_discovery_lcd.c
- BSP_LCD_DrawPixel() : stm32f769i_discovery_lcd.c
- BSP_LCD_DrawPolygon() : stm32f769i_discovery_lcd.c
- BSP_LCD_DrawRect() : stm32f769i_discovery_lcd.c
- BSP_LCD_DrawVLine() : stm32f769i_discovery_lcd.c
- BSP_LCD_FillCircle() : stm32f769i_discovery_lcd.c
- BSP_LCD_FillEllipse() : stm32f769i_discovery_lcd.c
- BSP_LCD_FillPolygon() : stm32f769i_discovery_lcd.c
- BSP_LCD_FillRect() : stm32f769i_discovery_lcd.c
- BSP_LCD_GetBackColor() : stm32f769i_discovery_lcd.c
- BSP_LCD_GetFont() : stm32f769i_discovery_lcd.c
- BSP_LCD_GetTextColor() : stm32f769i_discovery_lcd.c
- BSP_LCD_GetXSize() : stm32f769i_discovery_lcd.c
- BSP_LCD_GetYSize() : stm32f769i_discovery_lcd.c
- BSP_LCD_HDMIInitEx() : stm32f769i_discovery_lcd.h
- BSP_LCD_Init() : stm32f769i_discovery_lcd.c
- BSP_LCD_InitEx() : stm32f769i_discovery_lcd.c
- BSP_LCD_LayerDefaultInit() : stm32f769i_discovery_lcd.c
- BSP_LCD_MspDeInit() : stm32f769i_discovery_lcd.c
- BSP_LCD_MspInit() : stm32f769i_discovery_lcd.c
- BSP_LCD_ReadPixel() : stm32f769i_discovery_lcd.c
- BSP_LCD_Reset() : stm32f769i_discovery_lcd.c
- BSP_LCD_ResetColorKeying() : stm32f769i_discovery_lcd.c
- BSP_LCD_SelectLayer(): stm32f769i_discovery_lcd.c
- BSP_LCD_SetBackColor(): stm32f769i_discovery_lcd.c
- BSP_LCD_SetBrightness(): stm32f769i_discovery_lcd.c
- BSP_LCD_SetColorKeying(): stm32f769i_discovery_lcd.c
- BSP_LCD_SetFont(): stm32f769i_discovery_lcd.c
- BSP_LCD_SetLayerAddress(): stm32f769i_discovery_lcd.c
- BSP_LCD_SetLayerVisible(): stm32f769i_discovery_lcd.c
- BSP_LCD_SetLayerWindow(): stm32f769i_discovery_lcd.c
- BSP_LCD_SetTextColor(): stm32f769i_discovery_lcd.c
- BSP_LCD_SetTransparency(): stm32f769i_discovery_lcd.c
- BSP_LCD_SetXSize(): stm32f769i_discovery_lcd.c
- BSP_LCD_SetYSize(): stm32f769i_discovery_lcd.c
- BSP_LED_DeInit(): stm32f769i_discovery.c, stm32f769i_discovery.h
- BSP_LED_Init(): stm32f769i_discovery.c, stm32f769i_discovery.h
- BSP_LED_Off(): stm32f769i_discovery.c, stm32f769i_discovery.h
- BSP_LED_On(): stm32f769i_discovery.h, stm32f769i_discovery.c
- BSP_LED_Toggle(): stm32f769i_discovery.c, stm32f769i_discovery.h
- BSP_PB_DeInit(): stm32f769i_discovery.h, stm32f769i_discovery.c
- BSP_PB_GetState(): stm32f769i_discovery.c, stm32f769i_discovery.h
- BSP_PB_Init(): stm32f769i_discovery.c, stm32f769i_discovery.h
- BSP_QSPI_DeInit(): stm32f769i_discovery_qspi.c
- BSP_QSPI_EnableMemoryMappedMode(): stm32f769i_discovery_qspi.c
- BSP_QSPI_Erase_Block(): stm32f769i_discovery_qspi.c
- BSP_QSPI_Erase_Chip(): stm32f769i_discovery_qspi.c
- BSP_QSPI_GetInfo(): stm32f769i_discovery_qspi.c
- BSP_QSPI_GetStatus(): stm32f769i_discovery_qspi.c
- BSP_QSPI_Init(): stm32f769i_discovery_qspi.c
- BSP_QSPI_MspDeInit(): stm32f769i_discovery_qspi.h,
stm32f769i_discovery_qspi.c

- **BSP_QSPI_MspInit()**: `stm32f769i_discovery_qspi.c`, `stm32f769i_discovery_qspi.h`
- **BSP_QSPI_Read()**: `stm32f769i_discovery_qspi.c`
- **BSP_QSPI_Write()**: `stm32f769i_discovery_qspi.c`
- **BSP_SD_AbortCallback()**: `stm32f769i_discovery_sd.c`, `stm32f769i_discovery_sd.h`
- **BSP_SD_DeInit()**: `stm32f769i_discovery_sd.c`, `stm32f769i_discovery_sd.h`
- **BSP_SD_Detect_MspInit()**: `stm32f769i_discovery_sd.c`, `stm32f769i_discovery_sd.h`
- **BSP_SD_Erase()**: `stm32f769i_discovery_sd.c`, `stm32f769i_discovery_sd.h`
- **BSP_SD_GetCardInfo()**: `stm32f769i_discovery_sd.c`, `stm32f769i_discovery_sd.h`
- **BSP_SD_GetCardState()**: `stm32f769i_discovery_sd.c`, `stm32f769i_discovery_sd.h`
- **BSP_SD_Init()**: `stm32f769i_discovery_sd.c`, `stm32f769i_discovery_sd.h`
- **BSP_SD_IsDetected()**: `stm32f769i_discovery_sd.c`, `stm32f769i_discovery_sd.h`
- **BSP_SD_ITConfig()**: `stm32f769i_discovery_sd.c`, `stm32f769i_discovery_sd.h`
- **BSP_SD_MspDeInit()**: `stm32f769i_discovery_sd.c`, `stm32f769i_discovery_sd.h`
- **BSP_SD_MspInit()**: `stm32f769i_discovery_sd.c`, `stm32f769i_discovery_sd.h`
- **BSP_SD_ReadBlocks()**: `stm32f769i_discovery_sd.c`, `stm32f769i_discovery_sd.h`
- **BSP_SD_ReadBlocks_DMA()**: `stm32f769i_discovery_sd.h`, `stm32f769i_discovery_sd.c`
- **BSP_SD_ReadCpltCallback()**: `stm32f769i_discovery_sd.h`, `stm32f769i_discovery_sd.c`
- **BSP_SD_WriteBlocks()**: `stm32f769i_discovery_sd.c`, `stm32f769i_discovery_sd.h`
- **BSP_SD_WriteBlocks_DMA()**: `stm32f769i_discovery_sd.c`, `stm32f769i_discovery_sd.h`
• BSP_SD_WriteCpltCallback() : stm32f769i_discovery_sd.c ,
  stm32f769i_discovery_sd.h
• BSP_SDRAM_DeInit() : stm32f769i_discovery_sdram.c ,
  stm32f769i_discovery_sdram.h
• BSP_SDRAM_Init() : stm32f769i_discovery_sdram.c ,
  stm32f769i_discovery_sdram.h
• BSP_SDRAM_Initialization_sequence() :
  stm32f769i_discovery_sdram.c ,
  stm32f769i_discovery_sdram.h
• BSP_SDRAM_MspDelInit() : stm32f769i_discovery_sdram.c ,
  stm32f769i_discovery_sdram.h
• BSP_SDRAM_MspInit() : stm32f769i_discovery_sdram.h ,
  stm32f769i_discovery_sdram.c
• BSP_SDRAM_ReadData() : stm32f769i_discovery_sdram.h ,
  stm32f769i_discovery_sdram.c
• BSP_SDRAM_ReadData_DMA() :
  stm32f769i_discovery_sdram.c ,
  stm32f769i_discovery_sdram.h
• BSP_SDRAM_Sendcmd() : stm32f769i_discovery_sdram.h ,
  stm32f769i_discovery_sdram.c
• BSP_SDRAM_WriteData() : stm32f769i_discovery_sdram.c ,
  stm32f769i_discovery_sdram.h
• BSP_SDRAM_WriteData_DMA() :
  stm32f769i_discovery_sdram.h ,
  stm32f769i_discovery_sdram.c
• BSP_TS_GetState() : stm32f769i_discovery_ts.c ,
  stm32f769i_discovery_ts.h
• BSP_TS_Init() : stm32f769i_discovery_ts.c ,
  stm32f769i_discovery_ts.h
• BSP_TS_INT_MspInit() : stm32f769i_discovery_ts.c ,
  stm32f769i_discovery_ts.h
• BSP_TS_ITConfig() : stm32f769i_discovery_ts.h ,
  stm32f769i_discovery_ts.c
STM32F769I-Discovery BSP User Manual

- d -

- DFSDMx_ChannelMspDeInit() : stm32f769i_discovery_audio.c
- DFSDMx_ChannelMspInit() : stm32f769i_discovery_audio.c
- DFSDMx_DeInit() : stm32f769i_discovery_audio.c
- DFSDMx_FilterMspDeInit() : stm32f769i_discovery_audio.c
- DFSDMx_FilterMspInit() : stm32f769i_discovery_audio.c
- DFSDMx_Init() : stm32f769i_discovery_audio.c
- DrawChar() : stm32f769i_discovery_lcd.c
- DSI_IO_WriteCmd() : stm32f769i_discovery_lcd.c
STM32F769I-Discovery BSP User Manual

- e -

- EEPROM_IO_Init() : stm32f769i_discovery.c, stm32f769i_discovery_eeprom.h
- EEPROM_IO_IsDeviceReady() : stm32f769i_discovery_eeprom.h, stm32f769i_discovery.c
- EEPROM_IO_ReadData() : stm32f769i_discovery.c, stm32f769i_discovery_eeprom.h
- EEPROM_IO_WriteData() : stm32f769i_discovery_eeprom.h, stm32f769i_discovery.c

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>File List</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>Functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defines</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **f** -

- FillTriangle() : `stm32f769i_discovery_lcd.c`

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery BSP User Manual by [doxygen](http://www.stackexchange.com) 1.7.6.1
STM32F769I-Discovery BSP User Manual

- h -

- HAL_DFSDM_FilterRegConvCpltCallback() : \texttt{stm32f769i\_discovery\_audio.c}
- HAL_DFSDM_FilterRegConvHalfCpltCallback() : \texttt{stm32f769i\_discovery\_audio.c}
- HAL_SAI_ErrorCallback() : \texttt{stm32f769i\_discovery\_audio.c}
- HAL_SAI_RxCpltCallback() : \texttt{stm32f769i\_discovery\_audio.c}
- HAL_SAI_RxHalfCpltCallback() : \texttt{stm32f769i\_discovery\_audio.c}
- HAL_SAI_TxCpltCallback() : \texttt{stm32f769i\_discovery\_audio.c}
- HAL_SAI_TxHalfCpltCallback() : \texttt{stm32f769i\_discovery\_audio.c}
- HAL_SD_AbortCallback() : \texttt{stm32f769i\_discovery\_sd.c}
- HAL_SD_RxCpltCallback() : \texttt{stm32f769i\_discovery\_sd.c}
- HAL_SD_TxCpltCallback() : \texttt{stm32f769i\_discovery\_sd.c}
- HDMI_IO_Delay() : \texttt{stm32f769i\_discovery.c}
- HDMI_IO_Init() : \texttt{stm32f769i\_discovery.c}
- HDMI_IO_Read() : \texttt{stm32f769i\_discovery.c}
- HDMI_IO_Write() : \texttt{stm32f769i\_discovery.c}

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
- i -

- I2Cx_Error() : *stm32f769i_discovery.c*
- I2Cx_Init() : *stm32f769i_discovery.c*
- I2Cx_IsDeviceReady() : *stm32f769i_discovery.c*
- I2Cx_MspInit() : *stm32f769i_discovery.c*
- I2Cx_ReadMultiple() : *stm32f769i_discovery.c*
- I2Cx_WriteMultiple() : *stm32f769i_discovery.c*
STM32F769I-Discovery BSP User Manual

- I -

- LCD_IO_GetID() : stm32f769i_discovery_lcd.c
- LL_ConvertLineToARGB8888() : stm32f769i_discovery_lcd.c
- LL_FillBuffer() : stm32f769i_discovery_lcd.c

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
STM32F769I-Discovery BSP User Manual

- o -

  - OTM8009A_IO_Delay() : stm32f769i_discovery.c

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>File List</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>Functions</td>
<td>Variables</td>
<td>Typedefs</td>
</tr>
<tr>
<td>Defines</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- q -

- QSPI_AutoPollingMemReady() : `stm32f769i_discovery_qspi.c`
- QSPI_DummyCyclesCfg() : `stm32f769i_discovery_qspi.c`
- QSPI_EnterFourBytesAddress() : `stm32f769i_discovery_qspi.c`
- QSPI_EnterMemory_QPI() : `stm32f769i_discovery_qspi.c`
- QSPI_ExitMemory_QPI() : `stm32f769i_discovery_qspi.c`
- QSPI_OutDrvStrengthCfg() : `stm32f769i_discovery_qspi.c`
- QSPI_ResetMemory() : `stm32f769i_discovery_qspi.c`
- QSPI_WriteEnable() : `stm32f769i_discovery_qspi.c`

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
- s -

- **SAI_AUDIO_IN_MspDeInit()**: *stm32f769i_discovery_audio.c*
- **SAI_AUDIO_IN_MspInit()**: *stm32f769i_discovery_audio.c*
- **SAIx_In_DeInit()**: *stm32f769i_discovery_audio.c*
- **SAIx_In_Init()**: *stm32f769i_discovery_audio.c*
- **SAIx_Out_DeInit()**: *stm32f769i_discovery_audio.c*
- **SAIx_Out_Init()**: *stm32f769i_discovery_audio.c*
- t -

- TS_IO_Delay() : *stm32f769i_discovery.c*
- TS_IO_Init() : *stm32f769i_discovery.c*
- TS_IO_Read() : *stm32f769i_discovery.c*
- TS_IO_ReadMultiple() : *stm32f769i_discovery.c*
- TS_IO_Write() : *stm32f769i_discovery.c*
- TS_IO_WriteMultiple() : *stm32f769i_discovery.c*

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery BSP User Manual by [doxygen](http://www.stackoverflow.com) 1.7.6.1
STM32F769I-Discovery BSP User Manual

- a -
  - ActiveLayer : \texttt{stm32f769i\_discovery\_lcd.c}
  - AppBuffHalf : \texttt{stm32f769i\_discovery\_audio.c}
  - AppBuffTrigger : \texttt{stm32f769i\_discovery\_audio.c}
  - audio\_drv : \texttt{stm32f769i\_discovery\_audio.c}
  - AudioIn\_ChannelNumber : \texttt{stm32f769i\_discovery\_audio.c}
  - AudioIn\_Device : \texttt{stm32f769i\_discovery\_audio.c}

- b -
  - BUTTON\_IRQn : \texttt{stm32f769i\_discovery.c}
  - BUTTON\_PIN : \texttt{stm32f769i\_discovery.c}
  - BUTTON\_PORT : \texttt{stm32f769i\_discovery.c}

- c -
  - Command : \texttt{stm32f769i\_discovery\_sdram.c}

- d -
  - DmaButtomLeftRecCplt : \texttt{stm32f769i\_discovery\_audio.c}
  - DmaButtomLeftRecHalfCplt : \texttt{stm32f769i\_discovery\_audio.c}
- e -

- EEPROMAddress: stm32f769i_discovery_eeprom.c
- EEPROMDataRead: stm32f769i_discovery_eeprom.c
- EEPROMDataWrite: stm32f769i_discovery_eeprom.c

- g -

- GPIO_PIN: stm32f769i_discovery.c
- GPIO_PORT: stm32f769i_discovery.c

- h -

- haudio_in_sai: stm32f769i_discovery_audio.c
- haudio_out_sai: stm32f769i_discovery_audio.c
- hAudioIn: stm32f769i_discovery_audio.c
- hAudioInButtomLeftChannel: stm32f769i_discovery_audio.c
- hAudioInButtomLeftFilter: stm32f769i_discovery_audio.c
- hAudioInButtomRightChannel: stm32f769i_discovery_audio.c
- hAudioInButtomRightFilter: stm32f769i_discovery_audio.c
- hAudioInTopLeftChannel: stm32f769i_discovery_audio.c
- hAudioInTopLeftFilter: stm32f769i_discovery_audio.c
- hAudioInTopRightChannel: stm32f769i_discovery_audio.c
- hAudioInTopRightFilter: stm32f769i_discovery_audio.c
- hdma2d_discovery: stm32f769i_discovery_lcd.c, stm32f769i_discovery_lcd.h
- hDmaButtomLeft: stm32f769i_discovery_audio.c
- hDmaButtomRight: stm32f769i_discovery_audio.c
- hDmaTopLeft: stm32f769i_discovery_audio.c
- hDmaTopRight: stm32f769i_discovery_audio.c
- hdsi_discovery : stm32f769i_discovery_lcd.c
  hdsivideo_handle : stm32f769i_discovery_lcd.c
  hI2cAudioHandler : stm32f769i_discovery.c
  hI2cExtHandler : stm32f769i_discovery.c
  hltdc_discovery : stm32f769i_discovery_lcd.c
- i -
  I2C_Address : stm32f769i_discovery_ts.c
- l -
  lcd_x_size : stm32f769i_discovery_lcd.c
  lcd_y_size : stm32f769i_discovery_lcd.c
- p -
  pScratchBuff : stm32f769i_discovery_audio.c
- q -
  QSPIHandle : stm32f769i_discovery_qspi.c
- s -
  ScratchSize : stm32f769i_discovery_audio.c
  sdramHandle : stm32f769i_discovery_sdram.c
- t -
  Timing : stm32f769i_discovery_sdram.c
  ts_driver : stm32f769i_discovery_ts.c
  ts_event_string_tab : stm32f769i_discovery_ts.h ,
  stm32f769i_discovery_ts.c
  ts_gesture_id_string_tab : stm32f769i_discovery_ts.h ,
  stm32f769i_discovery_ts.c
  ts_orientation : stm32f769i_discovery_ts.c
- u -
STM32F769I-Discovery BSP User Manual

- pPoint: `stm32f769i_discovery_lcd.h`
STM32F769I-Discovery BSP User Manual

- Button_TypeDef : `stm32f769i_discovery.h`
- ButtonMode_TypeDef : `stm32f769i_discovery.h`
- ButtonValue_TypeDef : `stm32f769i_discovery.h`
- DISCO_Status_TypeDef : `stm32f769i_discovery.h`
- LCD_OrientationTypeDef : `stm32f769i_discovery_lcd.h`
- Led_TypeDef : `stm32f769i_discovery.h`
- Text_AlignModeTypdef : `stm32f769i_discovery_lcd.h`
- TS_GestureIdTypeDef : `stm32f769i_discovery_ts.h`
- TS_StatusTypeDef : `stm32f769i_discovery_ts.h`
- TS_TouchEventTypeDef : `stm32f769i_discovery_ts.h`

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
- b -
  - BUTTON_MODE_EXTI : stm32f769i_discovery.h
  - BUTTON_MODE_GPIO : stm32f769i_discovery.h
  - BUTTON_WAKEUP : stm32f769i_discovery.h

- c -
  - CENTER_MODE : stm32f769i_discovery_lcd.h

- d -
  - DISCO_ERROR : stm32f769i_discovery.h
  - DISCO_OK : stm32f769i_discovery.h

- g -
  - GEST_ID_MOVE_DOWN : stm32f769i_discovery_ts.h
  - GEST_ID_MOVE_LEFT : stm32f769i_discovery_ts.h
  - GEST_ID_MOVE_RIGHT : stm32f769i_discovery_ts.h
  - GEST_ID_MOVE_UP : stm32f769i_discovery_ts.h
  - GEST_ID_NB_MAX : stm32f769i_discovery_ts.h
  - GEST_ID_NO_GESTURE : stm32f769i_discovery_ts.h
- GEST_ID_ZOOM_IN : stm32f769i_discovery_ts.h
  GEST_ID_ZOOM_OUT : stm32f769i_discovery_ts.h
-
- LCD_ORIENTATION_INVALID : stm32f769i_discovery_lcd.h
  LCD_ORIENTATION_LANDSCAPE : stm32f769i_discovery_lcd.h
  LCD_ORIENTATION_PORTRAIT : stm32f769i_discovery_lcd.h
  LED1 : stm32f769i_discovery.h
  LED2 : stm32f769i_discovery.h
  LED_GREEN : stm32f769i_discovery.h
  LED_RED : stm32f769i_discovery.h
  LEFT_MODE : stm32f769i_discovery_lcd.h
-
- PB_RESET : stm32f769i_discovery.h
  PB_SET : stm32f769i_discovery.h
-
- RIGHT_MODE : stm32f769i_discovery_lcd.h
-
- TOUCH_EVENT_CONTACT : stm32f769i_discovery_ts.h
  TOUCH_EVENT_LIFT_UP : stm32f769i_discovery_ts.h
  TOUCH_EVENT_NB_MAX : stm32f769i_discovery_ts.h
  TOUCH_EVENT_NO_EVT : stm32f769i_discovery_ts.h
  TOUCH_EVENT_PRESS_DOWN : stm32f769i_discovery_ts.h
  TS_DEVICE_NOT_FOUND : stm32f769i_discovery_ts.h
  TS_ERROR : stm32f769i_discovery_ts.h
  TS_OK : stm32f769i_discovery_ts.h
  TS_TIMEOUT : stm32f769i_discovery_ts.h
STM32F769I-Discovery BSP User Manual

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1

- __DMAx_CLK_DISABLE : stm32f769i_discovery_sdram.h
- __DMAx_CLK_ENABLE : stm32f769i_discovery_sdram.h
- __DMAx_TxRx_CLK_ENABLE : stm32f769i_discovery_sd.h
- __STM32F769I_DISCOVERY_BSP_VERSION : stm32f769i_discovery.c
- __STM32F769I_DISCOVERY_BSP_VERSION_MAIN : stm32f769i_discovery.c
- __STM32F769I_DISCOVERY_BSP_VERSION_RC : stm32f769i_discovery.c
- __STM32F769I_DISCOVERY_BSP_VERSION_SUB1 : stm32f769i_discovery.c
- __STM32F769I_DISCOVERY_BSP_VERSION_SUB2 : stm32f769i_discovery.c
STM32F769I-Discovery BSP User Manual

- a -

- ABS : stm32f769i_discovery_lcd.c
- AUDIO_DFSDMx BUTTOM_LEFT_CHANNEL : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx BUTTOM_LEFT_FILTER : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx BUTTOM_RIGHT_CHANNEL : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx BUTTOM_RIGHT_FILTER : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx CKOUT_AF : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx CKOUT_DMIC_GPIO_CLK_ENABLE : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx CKOUT_DMIC_GPIO_PORT : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx CKOUT_PIN : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx CLK_ENABLE : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx BUTTOM_LEFT_IRQ : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx BUTTOM_LEFT_IRQHandler :
stm32f769i_discovery_audio.h

- AUDIO_DFSDMx_DMAx_BUTTOM_LEFT_STREAM : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_IRQ : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_IRQHandler : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_STREAM : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_CHANNEL : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_CLK_ENABLE : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_MEM_DATA_SIZE : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_PERIPH_DATA_SIZE : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_TOP_LEFT_IRQ : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_TOP_LEFT_IRQHandler : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_TOP_LEFT_STREAM : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_TOP_RIGHT_IRQ : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_TOP_RIGHT_IRQHandler : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMAx_TOP_RIGHT_STREAM : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMIC_DATIN1_PIN : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMIC_DATIN5_PIN : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMIC_DATIN_AF : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMIC_DATIN_GPIO_CLK_ENABLE : stm32f769i_discovery_audio.h
- AUDIO_DFSDMx_DMIC_DATIN_GPIO_PORT: `stm32f769i_discovery_audio.h`
- AUDIO_DFSDMx_TOP_LEFT_CHANNEL: `stm32f769i_discovery_audio.h`
- AUDIO_DFSDMx_TOP_LEFT_FILTER: `stm32f769i_discovery_audio.h`
- AUDIO_DFSDMx_TOP_RIGHT_CHANNEL: `stm32f769i_discovery_audio.h`
- AUDIO_DFSDMx_TOP_RIGHT_FILTER: `stm32f769i_discovery_audio.h`
- AUDIO_ERROR: `stm32f769i_discovery_audio.h`
- AUDIO_I2C_ADDRESS: `stm32f769i_discovery.h`
- AUDIO_IN_INT_GPIO_ENABLE: `stm32f769i_discovery_audio.h`
- AUDIO_IN_INT_GPIO_PIN: `stm32f769i_discovery_audio.h`
- AUDIO_IN_INT_GPIO_PORT: `stm32f769i_discovery_audio.h`
- AUDIO_IN_INT_IRQ: `stm32f769i_discovery_audio.h`
- AUDIO_IN_IRQ_PREPRIO: `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx: `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_AF: `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_CLK_DISABLE: `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_CLK_ENABLE: `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_DMAx_CHANNEL: `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_DMAx_CLK_ENABLE: `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_DMAx_IRQ: `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_DMAx_MEM_DATA_SIZE: `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_DMAx_PERIPH_DATA_SIZE: `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_DMAx_STREAM: `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_SD_ENABLE: `stm32f769i_discovery_audio.h`
- AUDIO_IN_SAIx_SD_GPIO_PORT: `stm32f769i_discovery_audio.h`
stm32f769i_discovery_audio.h
- AUDIO_IN_SAIx_SD_PIN : stm32f769i_discovery_audio.h
- AUDIO_OK : stm32f769i_discovery_audio.h
- AUDIO_OUT_IRQ_PREPRIORITY : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_AF : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_CLK_DISABLE : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_CLK_ENABLE : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_DMAx_CHANNEL : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_DMAx_CLK_ENABLE : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_DMAx_IRQ : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_DMAx_IRQHandler : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_DMAx_MEM_DATA_SIZE : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_DMAx_PERIPH_DATA_SIZE : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_DMAx_STREAM : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_FS_PIN : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_MCLK_ENABLE : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_MCLK_GPIO_PORT : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_MCLK_PIN : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_SCK_PIN : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_SD_FS_CLK_ENABLE : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_SD_FS_SCK_GPIO_PORT : stm32f769i_discovery_audio.h
- AUDIO_OUT_SAIx_SD_PIN : stm32f769i_discovery_audio.h
- AUDIO_TIMEOUT : stm32f769i_discovery_audio.h
- AUDIODATA_SIZE : stm32f769i_discovery_audio.h
- b -

- **BSP_AUDIO_FREQUENCY_11K**: `stm32f769i_discovery_audio.h`
- **BSP_AUDIO_FREQUENCY_16K**: `stm32f769i_discovery_audio.h`
- **BSP_AUDIO_FREQUENCY_22K**: `stm32f769i_discovery_audio.h`
- **BSP_AUDIO_FREQUENCY_32K**: `stm32f769i_discovery_audio.h`
- **BSP_AUDIO_FREQUENCY_44K**: `stm32f769i_discovery_audio.h`
- **BSP_AUDIO_FREQUENCY_48K**: `stm32f769i_discovery_audio.h`
- **BSP_AUDIO_FREQUENCY_8K**: `stm32f769i_discovery_audio.h`
- **BSP_AUDIO_FREQUENCY_96K**: `stm32f769i_discovery_audio.h`
- **BSP_LCD_DMA2D_IRQHandler**: `stm32f769i_discovery_lcd.h`
- **BSP_LCD_DSI_IRQHandler**: `stm32f769i_discovery_lcd.h`
- **BSP_LCD_LTDC_ER_IRQHandler**: `stm32f769i_discovery_lcd.h`
- **BSP_LCD_LTDC_IRQHandler**: `stm32f769i_discovery_lcd.h`
- **BSP_SD_CardInfo**: `stm32f769i_discovery_sd.h`
• BSP_SDMMC_DMA_Rx_IRQHandler : 
  `stm32f769i_discovery_sd.h`
• BSP_SDMMC_DMA_Tx_IRQHandler : 
  `stm32f769i_discovery_sd.h`
• BSP_SDMMC_IRQHandler : 
  `stm32f769i_discovery_sd.h`
• BSP_SDRAM_DMA_IRQHandler : 
  `stm32f769i_discovery_sdram.h`
• BUTTON_GPIO_CLK_ENABLE : `stm32f769i_discovery.h`
• BUTTON_USER : `stm32f769i_discovery.h`
• BUTTONn : `stm32f769i_discovery.h`
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>File List</td>
<td>Globals</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>All</th>
<th>Functions</th>
<th>Variables</th>
<th>Typedefs</th>
<th>Enumerations</th>
<th>Enumerator</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Defines</th>
</tr>
</thead>
<tbody>
<tr>
<td>_a b c d e h i l m o p q r s t u w</td>
</tr>
</tbody>
</table>

- `c`

- CODEC_AUDIOFRAME_SLOT_0123 :
  `stm32f769i_discovery_audio.h`
- CODEC_AUDIOFRAME_SLOT_02 :
  `stm32f769i_discovery_audio.h`
- CODEC_AUDIOFRAME_SLOT_13 :
  `stm32f769i_discovery_audio.h`

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery BSP User Manual by [doxygen](http://www.stackoverflow.com) 1.7.6.1
- d -

- DEFAULT_AUDIO_IN_BIT_RESOLUTION:
  `stm32f769i_discovery_audio.h`
- DEFAULT_AUDIO_IN_CHANNEL_NBR:
  `stm32f769i_discovery_audio.h`
- DEFAULT_AUDIO_IN_FREQ:
  `stm32f769i_discovery_audio.h`
- DEFAULT_AUDIO_IN_VOLUME:
  `stm32f769i_discovery_audio.h`
- DFSDM_CLOCK_DIVIDER:
  `stm32f769i_discovery_audio.c`
- DFSDM_FILTER_ORDER:
  `stm32f769i_discovery_audio.c`
- DFSDM_OVER_SAMPLING:
  `stm32f769i_discovery_audio.c`
- DFSDM_RIGHT_BIT_SHIFT:
  `stm32f769i_discovery_audio.c`
- DISCOVERY_AUDIO_I2Cx:
  `stm32f769i_discovery.h`
- DISCOVERY_AUDIO_I2Cx_CLK_ENABLE:
  `stm32f769i_discovery.h`
- DISCOVERY_AUDIO_I2Cx_ER_IRQn:
  `stm32f769i_discovery.h`
- DISCOVERY_AUDIO_I2Cx_EV_IRQn:
  `stm32f769i_discovery.h`
- DISCOVERY_AUDIO_I2Cx_FORCE_RESET:
  `stm32f769i_discovery.h`
- DISCOVERY_AUDIO_I2Cx_RELEASE_RESET:
  `stm32f769i_discovery.h`
• DISCOVERY_AUDIO_I2Cx_SCL_AF : stm32f769i_discovery.h
• DISCOVERY_AUDIO_I2Cx_SCL_GPIO_CLK_ENABLE : 
  stm32f769i_discovery.h
• DISCOVERY_AUDIO_I2Cx_SCL_GPIO_PORT : 
  stm32f769i_discovery.h
• DISCOVERY_AUDIO_I2Cx_SCL_PIN : stm32f769i_discovery.h
• DISCOVERY_AUDIO_I2Cx_SDA_AF : stm32f769i_discovery.h
• DISCOVERY_AUDIO_I2Cx_SDA_GPIO_CLK_ENABLE : 
  stm32f769i_discovery.h
• DISCOVERY_AUDIO_I2Cx_SDA_GPIO_PORT : 
  stm32f769i_discovery.h
• DISCOVERY_AUDIO_I2Cx_SDA_PIN : stm32f769i_discovery.h
• DISCOVERY_DMAx_CLK_ENABLE : 
  stm32f769i_discovery_audio.h
• DISCOVERY_EXT_I2Cx : stm32f769i_discovery.h
• DISCOVERY_EXT_I2Cx_CLK_ENABLE : 
  stm32f769i_discovery.h
• DISCOVERY_EXT_I2Cx_ER_IRQn : stm32f769i_discovery.h
• DISCOVERY_EXT_I2Cx_EV_IRQn : stm32f769i_discovery.h
• DISCOVERY_EXT_I2Cx_FORCE_RESET : 
  stm32f769i_discovery.h
• DISCOVERY_EXT_I2Cx_RELEASE_RESET : 
  stm32f769i_discovery.h
• DISCOVERY_EXT_I2Cx_SCL_PIN : stm32f769i_discovery.h
• DISCOVERY_EXT_I2Cx_SCL_SDA_AF : 
  stm32f769i_discovery.h
• DISCOVERY_EXT_I2Cx_SCL_SDA_GPIO_CLK_ENABLE : 
  stm32f769i_discovery.h
• DISCOVERY_EXT_I2Cx_SCL_SDA_GPIO_PORT : 
  stm32f769i_discovery.h
• DISCOVERY_EXT_I2Cx_SDA_PIN : stm32f769i_discovery.h
• DMA_MAX : stm32f769i_discovery_audio.h
• DMA_MAX_SZE : stm32f769i_discovery_audio.h
## EEPROM dames

- **EEPROM_FAIL** : `stm32f769i_discovery_eeprom.h`
- **EEPROM_I2C_ADDRESS_A01** : `stm32f769i_discovery.h`
- **EEPROM_I2C_ADDRESS_A02** : `stm32f769i_discovery.h`
- **EEPROM_MAX_SIZE** : `stm32f769i_discovery_eeprom.h`
- **EEPROM_MAX_TRIALS** : `stm32f769i_discovery_eeprom.h`
- **EEPROM_OK** : `stm32f769i_discovery_eeprom.h`
- **EEPROM_PAGENUM** : `stm32f769i_discovery_eeprom.h`
- **EEPROM_TIMEOUT** : `stm32f769i_discovery_eeprom.h`
- h -

- HDMI_ADV7533_ID : `stm32f769i_discovery_lcd.h`
- HDMI_FORMAT_720_480 : `stm32f769i_discovery_lcd.h`
- HDMI_FORMAT_720_576 : `stm32f769i_discovery_lcd.h`
STM32F769I-Discovery BSP User Manual

- i -

- INPUT_DEVICE_ANALOG_MIC : 
  
  stm32f769i_discovery_audio.h

- INPUT_DEVICE_DIGITAL_MIC : stm32f769i_discovery_audio.h

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
### LCD_Colors

- **LCD_COLOR_BLACK**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_BLUE**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_BROWN**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_CYAN**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_DARKBLUE**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_DARKCYAN**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_DARKGRAY**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_DARKGREEN**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_DARKMAGENTA**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_DARKRED**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_DARKYELLOW**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_GRAY**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_GREEN**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_LIGHTBLUE**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_LIGHTCYAN**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_LIGHTGRAY**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_LIGHTGREEN**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_LIGHTMAGENTA**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_LIGHTRED**: `stm32f769i_discovery_lcd.h`  
- **LCD_COLOR_LIGHTYELLOW**: `stm32f769i_discovery_lcd.h`
- LCD_COLOR_MAGENTA : stm32f769i_discovery_lcd.h
- LCD_COLOR_ORANGE : stm32f769i_discovery_lcd.h
- LCD_COLOR_RED : stm32f769i_discovery_lcd.h
- LCD_COLOR_TRANSPARENT : stm32f769i_discovery_lcd.h
- LCD_COLOR_WHITE : stm32f769i_discovery_lcd.h
- LCD_COLOR_YELLOW : stm32f769i_discovery_lcd.h
- LCD_DEFAULT_FONT : stm32f769i_discovery_lcd.h
- LCD_DSI_ADDRESS : stm32f769i_discovery_lcd.c
- LCD_DSI_ID : stm32f769i_discovery_lcd.c
- LCD_DSI_ID_REG : stm32f769i_discovery_lcd.c
- LCD_DSI_PIXEL_DATA_FMT_RBG565 : stm32f769i_discovery_lcd.h
- LCD_DSI_PIXEL_DATA_FMT_RBG888 : stm32f769i_discovery_lcd.h
- LCD_ERROR : stm32f769i_discovery_lcd.h
- LCD_FB_START_ADDRESS : stm32f769i_discovery_lcd.h
- LCD_LayerCfgTypeDef : stm32f769i_discovery_lcd.h
- LCD_OK : stm32f769i_discovery_lcd.h
- LCD_OTM8009A_ID : stm32f769i_discovery_lcd.h
- LCD_TIMEOUT : stm32f769i_discovery_lcd.h
- LED1_GPIO_PORT : stm32f769i_discovery.h
- LED1_PIN : stm32f769i_discovery.h
- LED2_GPIO_PORT : stm32f769i_discovery.h
- LED2_PIN : stm32f769i_discovery.h
- LEDn : stm32f769i_discovery.h
- LEDx_GPIO_CLK_DISABLE : stm32f769i_discovery.h
- LEDx_GPIO_CLK_ENABLE : stm32f769i_discovery.h
- LTDC_ACTIVE_LAYER_BACKGROUND : stm32f769i_discovery_lcd.h
- LTDC_ACTIVE_LAYER_FOREGROUND : stm32f769i_discovery_lcd.h
- LTDC_DEFAULT_ACTIVE_LAYER : stm32f769i_discovery_lcd.h
- LTDC_MAX_LAYER_NUMBER : stm32f769i_discovery_lcd.h
- LTDC_NB_OFLAYERS : stm32f769i_discovery_lcd.h

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery
- m -

- MSD_ERROR : `stm32f769i_discovery_sd.h`
- MSD_ERROR_SD_NOT_PRESENT : `stm32f769i_discovery_sd.h`
- MSD_OK : `stm32f769i_discovery_sd.h`
- o -

- OTG_HS_OVER_CURRENT_PIN : `stm32f769i_discovery.h`
- OTG_HS_OVER_CURRENT_PORT : `stm32f769i_discovery.h`
- OTG_HS_OVER_CURRENT_PORT_CLK_ENABLE : `stm32f769i_discovery.h`
- OUTPUT_DEVICE_HEADPHONE1 : `stm32f769i_discovery_audio.h`
- OUTPUT_DEVICE_HEADPHONE2 : `stm32f769i_discovery_audio.h`
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>File List</strong></td>
<td><strong>Globals</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>Functions</td>
<td>Variables</td>
<td>Typedefs</td>
</tr>
</tbody>
</table>

- **p** -
  - POLY_X : *stm32f769i_discovery_lcd.c*
  - POLY_Y : *stm32f769i_discovery_lcd.c*

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery BSP User Manual by [doxygen](https://www.doxygen.org) 1.7.6.1
STM32F769I-Discovery BSP User Manual

- q -

- QSPI_BUSY : stm32f769i_discovery_qspi.h
- QSPI_CLK_DISABLE : stm32f769i_discovery_qspi.h
- QSPI_CLK_ENABLE : stm32f769i_discovery_qspi.h
- QSPI_CLK_GPIO_CLK_ENABLE : stm32f769i_discovery_qspi.h
- QSPI_CLK_GPIO_PORT : stm32f769i_discovery_qspi.h
- QSPI_CLK_PIN : stm32f769i_discovery_qspi.h
- QSPI_CLK_PIN_AF : stm32f769i_discovery_qspi.h
- QSPI_CS_GPIO_CLK_ENABLE : stm32f769i_discovery_qspi.h
- QSPI_CS_GPIO_PORT : stm32f769i_discovery_qspi.h
- QSPI_CS_PIN : stm32f769i_discovery_qspi.h
- QSPI_CS_PIN_AF : stm32f769i_discovery_qspi.h
- QSPI_D0_GPIO_CLK_ENABLE : stm32f769i_discovery_qspi.h
- QSPI_D0_GPIO_PORT : stm32f769i_discovery_qspi.h
- QSPI_D0_PIN : stm32f769i_discovery_qspi.h
- QSPI_D0_PIN_AF : stm32f769i_discovery_qspi.h
- QSPI_D1_GPIO_CLK_ENABLE : stm32f769i_discovery_qspi.h
- QSPI_D1_GPIO_PORT : stm32f769i_discovery_qspi.h
- QSPI_D1_PIN : stm32f769i_discovery_qspi.h
- QSPI_D1_PIN_AF : stm32f769i_discovery_qspi.h
- QSPI_D2_GPIO_CLK_ENABLE : stm32f769i_discovery_qspi.h
- QSPI_D2_GPIO_PORT : `stm32f769i_discovery_qspi.h`
- QSPI_D2_PIN : `stm32f769i_discovery_qspi.h`
- QSPI_D2_PIN_AF : `stm32f769i_discovery_qspi.h`
- QSPI_D3_GPIO_CLK_ENABLE : `stm32f769i_discovery_qspi.h`
- QSPI_D3_GPIO_PORT : `stm32f769i_discovery_qspi.h`
- QSPI_D3_PIN : `stm32f769i_discovery_qspi.h`
- QSPI_D3_PIN_AF : `stm32f769i_discovery_qspi.h`
- QSPI_ERROR : `stm32f769i_discovery_qspi.h`
- QSPI_FORCE_RESET : `stm32f769i_discovery_qspi.h`
- QSPI_NOT_SUPPORTED : `stm32f769i_discovery_qspi.h`
- QSPI_OK : `stm32f769i_discovery_qspi.h`
- QSPI_RELEASE_RESET : `stm32f769i_discovery_qspi.h`
- QSPI_SUSPENDED : `stm32f769i_discovery_qspi.h`
## STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>File List</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>Functions</td>
<td>Variables</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defines</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **r** -

- **REFRESH_COUNT**: `stm32f769i_discovery_sdram.h`

---

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery BSP User Manual by [doxygen](https://www.doxygen.org/) 1.7.6.1
STM32F769I-Discovery BSP User Manual

- s -

- SaturaLH : stm32f769i_discovery_audio.c
- SD_DATATIMEOUT : stm32f769i_discovery_sd.h
- SD_DETECT_EXTI_IRQn : stm32f769i_discovery.h
- SD_DETECT_GPIO_CLK_DISABLE : stm32f769i_discovery.h
- SD_DETECT_GPIO_CLK_ENABLE : stm32f769i_discovery.h
- SD_DETECT_GPIO_PORT : stm32f769i_discovery.h
- SD_DETECT_PIN : stm32f769i_discovery.h
- SD_DetectIRQHandler : stm32f769i_discovery_sd.h
- SD_DMAx_Rx_CHANNEL : stm32f769i_discovery_sd.h
- SD_DMAx_Rx_IRQn : stm32f769i_discovery_sd.h
- SD_DMAx_Rx_STREAM : stm32f769i_discovery_sd.h
- SD_DMAx_Tx_CHANNEL : stm32f769i_discovery_sd.h
- SD_DMAx_Tx_IRQn : stm32f769i_discovery_sd.h
- SD_DMAx_Tx_STREAM : stm32f769i_discovery_sd.h
- SD_NOT_PRESENT : stm32f769i_discovery_sd.h
- SD_PRESENT : stm32f769i_discovery_sd.h
- SD_TRANSFER_BUSY : stm32f769i_discovery_sd.h
- SD_TRANSFER_OK : stm32f769i_discovery_sd.h
- SDCLOCK_PERIOD : stm32f769i_discovery_sdram.h
- SDRAM_DEVICE_ADDR : stm32f769i_discovery_sdram.h
- SDRAM_DEVICE_SIZE : stm32f769i_discovery_sdram.h
- SDRAM_DMAx_CHANNEL : stm32f769i_discovery_sdram.h
- SDRAM_DMAx_IRQn : stm32f769i_discovery_sdram.h
- SDRAM_DMAx_STREAM : stm32f769i_discovery_sdram.h
- SDRAM_ERROR : stm32f769i_discovery_sdram.h
- SDRAM_MEMORY_WIDTH : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_BURST_LENGTH_1 : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_BURST_LENGTH_2 : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_BURST_LENGTH_4 : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_BURST_LENGTH_8 : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_BURST_TYPE_INTERLEAVED : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_CAS_LATENCY_2 : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_CAS_LATENCY_3 : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_OPERATING_MODE_STANDARD : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_WRITEBURST_MODE_PROGRAMMED : stm32f769i_discovery_sdram.h
- SDRAM_MODEREG_WRITEBURST_MODE_SINGLE : stm32f769i_discovery_sdram.h
- SDRAM_OK : stm32f769i_discovery_sdram.h
- SDRAM_TIMEOUT : stm32f769i_discovery_sdram.h
- t -

- TS_I2C_ADDRESS : stm32f769i_discovery.h
- TS_I2C_ADDRESS_A02 : stm32f769i_discovery.h
- TS_INT_EXTI_IRQn : stm32f769i_discovery.h
- TS_INT_GPIO_CLK_DISABLE : stm32f769i_discovery.h
- TS_INT_GPIO_CLK_ENABLE : stm32f769i_discovery.h
- TS_INT_GPIO_PORT : stm32f769i_discovery.h
- TS_INT_PIN : stm32f769i_discovery.h
- TS_IRQ_PENDING : stm32f769i_discovery_ts.h
- TS_MAX_NB_TOUCH : stm32f769i_discovery_ts.h
- TS_NO_IRQ_PENDING : stm32f769i_discovery_ts.h
- TS_SWAP_NONE : stm32f769i_discovery_ts.h
- TS_SWAP_X : stm32f769i_discovery_ts.h
- TS_SWAP_XY : stm32f769i_discovery_ts.h
- TS_SWAP_Y : stm32f769i_discovery_ts.h

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery BSP User Manual by **doxygen** 1.7.6.1
- u -

- USER_BUTTON_EXTI_IRQn : `stm32f769i_discovery.h`
- USER_BUTTON_GPIO_CLK_DISABLE : `stm32f769i_discovery.h`
- USER_BUTTON_GPIO_CLK_ENABLE : `stm32f769i_discovery.h`
- USER_BUTTON_GPIO_PORT : `stm32f769i_discovery.h`
- USER_BUTTON_PIN : `stm32f769i_discovery.h`
STM32F769I-Discovery BSP User Manual

- W -

- WAKEUP_BUTTON_EXTIIRQn : `stm32f769i_discovery.h`
- WAKEUP_BUTTON_GPIO_CLK_DISABLE : `stm32f769i_discovery.h`
- WAKEUP_BUTTON_GPIO_CLK_ENABLE : `stm32f769i_discovery.h`
- WAKEUP_BUTTON_GPIO_PORT : `stm32f769i_discovery.h`
- WAKEUP_BUTTON_PIN : `stm32f769i_discovery.h`
STM32F769I-Discovery BSP User Manual

stm32f769i_discovery.c File Reference

This file provides a set of firmware functions to manage LEDs, push-buttons, external SDRAM, external QSPI Flash, RF EEPROM, available on STM32F769I-Discovery board (MB1225) from STMicroelectronics. More...

#include "stm32f769i_discovery.h"

Go to the source code of this file.
Defines

<table>
<thead>
<tr>
<th>Define</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__STM32F769I_DISCOVERY_BSP_VERSION_MAIN</code></td>
<td>0x02</td>
</tr>
<tr>
<td>STM32F769I Discovery BSP Driver version number V2.0.0.</td>
<td></td>
</tr>
<tr>
<td><code>__STM32F769I_DISCOVERY_BSP_VERSION_SUB1</code></td>
<td>0x00</td>
</tr>
<tr>
<td><code>__STM32F769I_DISCOVERY_BSP_VERSION_SUB2</code></td>
<td>0x00</td>
</tr>
<tr>
<td><code>__STM32F769I_DISCOVERY_BSP_VERSION_RC</code></td>
<td>0x00</td>
</tr>
<tr>
<td><code>__STM32F769I_DISCOVERY_BSP_VERSION</code></td>
<td></td>
</tr>
</tbody>
</table>
## Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td><strong>I2Cx_MspInit</strong> (I2C_HandleTypeDef *i2c_handler)</td>
<td>Initializes I2C MSP.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>I2Cx_Init</strong> (I2C_HandleTypeDef *i2c_handler)</td>
<td>Initializes I2C HAL.</td>
</tr>
<tr>
<td>static HAL_StatusTypeDef</td>
<td><strong>I2Cx_ReadMultiple</strong> (I2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t *Buffer, uint16_t Length)</td>
<td>Reads multiple data.</td>
</tr>
<tr>
<td>static HAL_StatusTypeDef</td>
<td><strong>I2Cx_WriteMultiple</strong> (I2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t *Buffer, uint16_t Length)</td>
<td>Writes a value in a register of the device through BUS in using DMA mode.</td>
</tr>
<tr>
<td>static HAL_StatusTypeDef</td>
<td><strong>I2Cx_IsDeviceReady</strong> (I2C_HandleTypeDef *i2c_handler, uint16_t DevAddress, uint32_t Trials)</td>
<td>Checks if target device is ready for communication.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>I2Cx_Error</strong> (I2C_HandleTypeDef *i2c_handler, uint8_t Addr)</td>
<td>Manages error callback by re-initializing I2C.</td>
</tr>
<tr>
<td>void</td>
<td><strong>AUDIO_IO_Init</strong> (void)</td>
<td>Initializes Audio low level.</td>
</tr>
<tr>
<td>void</td>
<td><strong>AUDIO_IO_DeInit</strong> (void)</td>
<td>DeInitializes Audio low level.</td>
</tr>
<tr>
<td>void</td>
<td><strong>AUDIO_IO_Write</strong> (uint8_t Addr, uint16_t Reg, uint16_t Value)</td>
<td></td>
</tr>
</tbody>
</table>
### Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>uint16_t</strong> AUDIO_IO_Read (uint8_t Addr, uint16_t Reg)</td>
<td>Reads a single data.</td>
</tr>
<tr>
<td><strong>void</strong> AUDIO_IO_Delay (uint32_t Delay)</td>
<td>AUDIO Codec delay.</td>
</tr>
<tr>
<td><strong>void</strong> HDMI_IO_Init (void)</td>
<td>Initializes HDMI IO low level.</td>
</tr>
<tr>
<td><strong>void</strong> HDMI_IO_Delay (uint32_t Delay)</td>
<td>HDMI delay.</td>
</tr>
<tr>
<td><strong>void</strong> HDMI_IO_Write (uint8_t Addr, uint8_t Reg, uint8_t Value)</td>
<td>HDMI writes single data.</td>
</tr>
<tr>
<td><strong>uint8_t</strong> HDMI_IO_Read (uint8_t Addr, uint8_t Reg)</td>
<td>Reads single data with I2C communication channel from HDMI bridge.</td>
</tr>
<tr>
<td><strong>void</strong> EEPROM_IO_Init (void)</td>
<td>Initializes peripherals used by the I2C EEPROM driver.</td>
</tr>
<tr>
<td><strong>HAL_StatusTypeDef</strong> EEPROM_IO_WriteData (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize)</td>
<td>Write data to I2C EEPROM driver in using DMA channel.</td>
</tr>
<tr>
<td><strong>HAL_StatusTypeDef</strong> EEPROM_IO_ReadData (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize)</td>
<td>Read data from I2C EEPROM driver in using DMA channel.</td>
</tr>
<tr>
<td><strong>HAL_StatusTypeDef</strong> EEPROM_IO_IsDeviceReady (uint16_t DevAddress, uint32_t Trials)</td>
<td>Checks if target device is ready for communication.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void TS_IO_Init (void)</code></td>
<td>Initializes Touchscreen low level.</td>
</tr>
<tr>
<td><code>void TS_IO_Write (uint8_t Addr, uint8_t Reg, uint8_t Value)</code></td>
<td>Writes a single data.</td>
</tr>
<tr>
<td><code>uint8_t TS_IO_Read (uint8_t Addr, uint8_t Reg)</code></td>
<td>Reads a single data.</td>
</tr>
<tr>
<td><code>uint16_t TS_IO_ReadMultiple (uint8_t Addr, uint8_t Reg, uint8_t *Buffer, uint16_t Length)</code></td>
<td>Reads multiple data with I2C communication channel from TouchScreen.</td>
</tr>
<tr>
<td><code>void TS_IO_WriteMultiple (uint8_t Addr, uint8_t Reg, uint8_t *Buffer, uint16_t Length)</code></td>
<td>Writes multiple data with I2C communication channel from MCU to TouchScreen.</td>
</tr>
<tr>
<td><code>void TS_IO_Delay (uint32_t Delay)</code></td>
<td>Delay function used in TouchScreen low level driver.</td>
</tr>
<tr>
<td><code>void OTM8009A_IO_Delay (uint32_t Delay)</code></td>
<td>OTM8009A delay.</td>
</tr>
<tr>
<td><code>uint32_t BSP_GetVersion (void)</code></td>
<td>This method returns the STM32F769I Discovery BSP Driver revision.</td>
</tr>
<tr>
<td><code>void BSP_LED_Init (Led_TypeDef Led)</code></td>
<td>Configures LED GPIO.</td>
</tr>
<tr>
<td><code>void BSP_LED_DeInit (Led_TypeDef Led)</code></td>
<td>DeInit LEDs.</td>
</tr>
<tr>
<td><code>void BSP_LED_On (Led_TypeDef Led)</code></td>
<td>Turns selected LED On.</td>
</tr>
<tr>
<td><code>void BSP_LED_Off (Led_TypeDef Led)</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>void BSP_LED_Toggle (Led_TypeDef Led)</code></td>
<td>Toggles the selected LED.</td>
</tr>
<tr>
<td><code>void BSP_PB_Init (Button_TypeDef Button, ButtonMode_TypeDef Button_Mode)</code></td>
<td>Configures button GPIO and EXTI Line.</td>
</tr>
<tr>
<td><code>void BSP_PB_DeInit (Button_TypeDef Button)</code></td>
<td>Push Button DeInit.</td>
</tr>
<tr>
<td><code>uint32_t BSP_PB_GetState (Button_TypeDef Button)</code></td>
<td>Returns the selected button state.</td>
</tr>
</tbody>
</table>
### Variables

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uint32_t</code></td>
<td><code>GPIO_PIN</code></td>
<td>[LEDn]</td>
</tr>
<tr>
<td><code>GPIO_TypeDef *</code></td>
<td><code>GPIO_PORT</code></td>
<td>[LEDn]</td>
</tr>
<tr>
<td><code>GPIO_TypeDef *</code></td>
<td><code>BUTTON_PORT</code></td>
<td>[BUTTONn] = {WAKEUP_BUTTON_GPIO_PORT}</td>
</tr>
<tr>
<td><code>const uint16_t</code></td>
<td><code>BUTTON_PIN</code></td>
<td>[BUTTONn] = {WAKEUP_BUTTON_PIN}</td>
</tr>
<tr>
<td><code>const uint16_t</code></td>
<td><code>BUTTON_IRQn</code></td>
<td>[BUTTONn] = {WAKEUP_BUTTON_EXTI_IRQn}</td>
</tr>
<tr>
<td><code>static I2C_HandleTypeDef</code></td>
<td><code>hI2cAudioHandler</code></td>
<td>= {0}</td>
</tr>
<tr>
<td><code>static I2C_HandleTypeDef</code></td>
<td><code>hI2cExtHandler</code></td>
<td>= {0}</td>
</tr>
</tbody>
</table>
Detailed Description

This file provides a set of firmware functions to manage LEDs, push-buttons, external SDRAM, external QSPI Flash, RF EEPROM, available on STM32F769I-Discovery board (MB1225) from STMicroelectronics.

Author:
   MCD Application Team

Version:
   V2.0.0

Date:
   30-December-2016

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

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

Definition in file stm32f769i_discovery.c.
STM32F769I-Discovery BSP User Manual

stm32f769i_discovery.h File Reference

This file contains definitions for STM32F769I-Discovery LEDs, push-buttons hardware resources. More...

#include "stm32f7xx_hal.h"

Go to the source code of this file.
Defines

```c
#define BUTTON_USER BUTTON_WAKEUP
#define LEDn ((uint8_t)2)
#define LED1_GPIO_PORT ((GPIO_TypeDef*)GPIOJ)
#define LED2_GPIO_PORT ((GPIO_TypeDef*)GPIOJ)
#define LEDx_GPIO_CLK_ENABLE() __HAL_RCC_GPIOJ_CLK_ENABLE()
#define LEDx_GPIO_CLK_DISABLE() __HAL_RCC_GPIOJ_CLK_DISABLE()
#define LED1_PIN ((uint32_t)GPIO_PIN_13)
#define LED2_PIN ((uint32_t)GPIO_PIN_5)
#define BUTTONn ((uint8_t)1)
#define WAKEUP_BUTTON_PIN GPIO_PIN_0
#define WAKEUP_BUTTON_GPIO_PORT GPIOA
#define WAKEUP_BUTTON_GPIO_CLK_ENABLE() __HAL_RCC_GPIOA_CLK_ENABLE()
#define WAKEUP_BUTTON_GPIO_CLK_DISABLE() __HAL_RCC_GPIOA_CLK_DISABLE()
#define WAKEUP_BUTTON_EXTI_IRQn EXTI0_IRQn
#define USER_BUTTON_PIN WAKEUP_BUTTON_PIN
#define USER_BUTTON_GPIO_PORT WAKEUP_BUTTON_GPIO_PORT
#define USER_BUTTON_GPIO_CLK_ENABLE() WAKEUP_BUTTON_GPIO_CLK_ENABLE()
#define USER_BUTTON_GPIO_CLK_DISABLE() WAKEUP_BUTTON_GPIO_CLK_DISABLE()
#define USER_BUTTON_EXTI_IRQn WAKEUP_BUTTON_EXTI_IRQn
#define OTG_HS_OVER_CURRENT_PIN GPIO_PIN_4
#define OTG_HS_OVER_CURRENT_PORT GPIOD
#define OTG_HS_OVER_CURRENT_PORT_CLK_ENABLE() __HAL_RCC_GPIOD_CLK_ENABLE()
#define SD_DETECT_PIN ((uint32_t)GPIO_PIN_15)
#define SD_DETECT_GPIO_PORT ((GPIO_TypeDef*)GPIOI)
#define SD_DETECT_GPIO_CLK_ENABLE() __HAL_RCC_GPIOI_CLK_ENABLE()
#define SD_DETECT_GPIO_CLK_DISABLE() __HAL_RCC_GPIOI_CLK_DISABLE()
#define SD_DETECT_EXTI_IRQn EXTI15_10_IRQn
```
#define TS_INT_PIN ((uint32_t)GPIO_PIN_13)
Touch screen interrupt signal.

#define TS_INT_GPIO_PORT ((GPIO_TypeDef*)GPIOI)
#define TS_INT_GPIO_CLK_ENABLE() __HAL_RCC_GPIOI_CLK_ENABLE()
#define TS_INT_GPIO_CLK_DISABLE() __HAL_RCC_GPIOI_CLK_DISABLE()
#define TS_INT_EXTI_IRQn EXTI15_10_IRQn
#define TS_I2C_ADDRESS ((uint16_t)0x54)
TouchScreen FT6206 Slave I2C address 1.

#define TS_I2C_ADDRESS_A02 ((uint16_t)0x70)
TouchScreen FT6336G Slave I2C address 2.

#define AUDIO_I2C_ADDRESS ((uint16_t)0x34)
Audio I2C Slave address.

#define EEPROM_I2C_ADDRESS_A01 ((uint16_t)0xA0)
EEPROM I2C Slave address 1.

#define EEPROM_I2C_ADDRESS_A02 ((uint16_t)0xA6)
EEPROM I2C Slave address 2.

#define DISCOVERY_AUDIO_I2Cx I2C4
User can use this section to tailor I2C4/I2C4 instance used and associated resources (audio codec).

#define DISCOVERY_AUDIO_I2Cx_CLK_ENABLE() __HAL_RCC_I2C4_CLK_ENABLE()
#define DISCOVERY_AUDIO_I2Cx_SCL_GPIO_CLK_ENABLE() __HAL_RCC_GPIOD_CLK_ENABLE()
#define DISCOVERY_AUDIO_I2Cx_SDA_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_ENABLE()
#define DISCOVERY_AUDIO_I2Cx_FORCE_RESET() __HAL_RCC_I2C4_FORCE_RESET()
#define DISCOVERY_AUDIO_I2Cx_RELEASE_RESET() __HAL_RCC_I2C4_RELEASE_RESET()
#define DISCOVERY_AUDIO_I2Cx_SCL_PIN GPIO_PIN_12
Definition for I2C4 Pins.

#define DISCOVERY_AUDIO_I2Cx_SCL_AF GPIO_AF4_I2C4
#define DISCOVERY_AUDIO_I2Cx_SCL_GPIO_PORT GPIOD
#define DISCOVERY_AUDIO_I2Cx_SDA_PIN GPIO_PIN_7
#define DISCOVERY_AUDIO_I2Cx_SDA_AF GPIO_AF11_I2C4
#define DISCOVERY_AUDIO_I2Cx_SDA_GPIO_PORT GPIOB
#define DISCOVERY_AUDIO_I2Cx_EV_IRQn I2C4_EV_IRQn
Definition of I2C4 interrupt requests.

#define DISCOVERY_AUDIO_I2Cx_ER_IRQn I2C4_ER_IRQn
#define DISCOVERY_EXT_I2Cx I2C1
User can use this section to tailor I2C1/I2C1 instance used and associated resources.

```c
#define DISCOVERY_EXT_I2C1_CLK_ENABLE() __HAL_RCC_I2C1_CLK_ENABLE()
#define DISCOVERY_DMA1_CLK_ENABLE() __HAL_RCC_DMA1_CLK_ENABLE()
#define DISCOVERY_EXT_I2C1_SCL_SDA_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_ENABLE()
#define DISCOVERY_EXT_I2C1_FORCE_RESET() __HAL_RCC_I2C1_FORCE_RESET()
#define DISCOVERY_EXT_I2C1_RELEASE_RESET() __HAL_RCC_I2C1_RELEASE_RESET()
#define DISCOVERY_EXT_I2C1_SCL_PIN GPIO_PIN_8
#define DISCOVERY_EXT_I2C1_SCL_SDA_GPIO_PORT GPIOB
#define DISCOVERY_EXT_I2C1_SCL_SDA_AF GPIO_AF4_I2C1
#define DISCOVERY_EXT_I2C1_SDA_PIN GPIO_PIN_9
#define DISCOVERY_EXT_I2C1_EV_IRQn I2C1_EV_IRQn
#define DISCOVERY_EXT_I2C1_ER_IRQn I2C1_ER_IRQn
```

Definition for I2C1 Pins.

```c
#define DISCOVERY_EXT_I2C1_SCL_SDA_GPIO_PORT GPIOB
#define DISCOVERY_EXT_I2C1_SCL_SDA_AF GPIO_AF4_I2C1
#define DISCOVERY_EXT_I2C1_SDA_PIN GPIO_PIN_9
#define DISCOVERY_EXT_I2C1_EV_IRQn I2C1_EV_IRQn
#define DISCOVERY_EXT_I2C1_ER_IRQn I2C1_ER_IRQn
```

Definition of I2C interrupt requests.
### Enumerations

<table>
<thead>
<tr>
<th>Enum Name</th>
<th>Definition</th>
<th>Board/Definition</th>
<th>More...</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Led_TypeDef</code></td>
<td><code>enum</code> { <code>LED1 = 0</code>, <code>LED_RED = LED1</code>, <code>LED2 = 1</code>, <code>LED_GREEN = LED2</code> }</td>
<td>Define for STM32F769I_DISCOVERY board.</td>
<td>More...</td>
</tr>
<tr>
<td><code>Button_TypeDef</code></td>
<td><code>enum</code> { <code>BUTTON_WAKEUP = 0</code> }</td>
<td>Button_TypeDef STM32F769I_DISCOVERY board Buttons definitions.</td>
<td>More...</td>
</tr>
<tr>
<td><code>ButtonMode_TypeDef</code></td>
<td><code>enum</code> { <code>BUTTON_MODE_GPIO = 0</code>, <code>BUTTON_MODE_EXTI = 1</code> }</td>
<td>ButtonMode_TypeDef STM32F769I_DISCOVERY board Buttons Modes definitions.</td>
<td>More...</td>
</tr>
<tr>
<td><code>ButtonValue_TypeDef</code></td>
<td><code>enum</code> { <code>PB_SET = 0</code>, <code>PB_RESET = !PB_SET</code> }</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>DISCO_Status_TypeDef</code></td>
<td><code>enum</code> { <code>DISCO_OK = 0</code>, <code>DISCO_ERROR = 1</code> }</td>
<td>DISCO_Status_TypeDef STM32F769I_DISCO board Status return possible values.</td>
<td>More...</td>
</tr>
</tbody>
</table>
## Functions

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uint32_t</code></td>
<td><code>BSP_GetVersion (void)</code></td>
<td>This method returns the STM32F769I Discovery BSP Driver revision.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP(LED_TypeDef Led)</code></td>
<td>Configures LED GPIO.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_LED_DeInit (Led_TypeDef Led)</code></td>
<td>DeInit LEDs.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_LED_On (Led_TypeDef Led)</code></td>
<td>Turns selected LED On.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_LED_Off (Led_TypeDef Led)</code></td>
<td>Turns selected LED Off.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_LED_Toggle (Led_TypeDef Led)</code></td>
<td>Toggles the selected LED.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_PB_Init (Button_TypeDef Button, ButtonMode_TypeDef Button_Mode)</code></td>
<td>Configures button GPIO and EXTI Line.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_PB_DeInit (Button_TypeDef Button)</code></td>
<td>Push Button DeInit.</td>
</tr>
<tr>
<td><code>uint32_t</code></td>
<td><code>BSP_PB_GetState (Button_TypeDef Button)</code></td>
<td>Returns the selected button state.</td>
</tr>
</tbody>
</table>
Detailed Description

This file contains definitions for STM32F769I-Discovery LEDs, push-buttons hardware resources.

Author:
    MCD Application Team

Version:
    V2.0.0

Date:
    30-December-2016

Attention:
© COPYRIGHT(c) 2016 STMicroelectronics

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

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

Definition in file stm32f769i_discovery.h.
This file provides the Audio driver for the STM32F769I-DISCOVERY board. More...

#include "stm32f769i_discovery_audio.h"

Go to the source code of this file.
Data Structures

```c
struct AUDIOIN_TypeDef
```
Defines

```c
#define DFSDM_OVER_SAMPLING(__FREQUENCY__)  
#define DFSDM_CLOCK_DIVIDER(__FREQUENCY__)  
#define DFSDM_FILTER_ORDER(__FREQUENCY__)   
#define DFSDM_RIGHT_BIT_SHIFT(__FREQUENCY__)  
#define SaturaLH(N, L, H) (((N)<(L))?(L):(((N)>(H))?(H):((N)))
```
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static void SAIx_Out_Init (uint32_t AudioFreq)</code></td>
<td>Initializes the Audio Codec audio interface (SAI).</td>
</tr>
<tr>
<td><code>static void SAIx_Out_DeInit (void)</code></td>
<td>Deinitializes the Audio Codec audio interface (SAI).</td>
</tr>
<tr>
<td><code>static void SAI_AUDIO_IN_MspInit (SAI_HandleTypeDef *hsai, void *Params)</code></td>
<td>Initializes SAI Audio IN MSP.</td>
</tr>
<tr>
<td><code>static void SAI_AUDIO_IN_MspDeInit (SAI_HandleTypeDef *hsai, void *Params)</code></td>
<td>De-Initializes SAI Audio IN MSP.</td>
</tr>
<tr>
<td><code>static void SAIx_In_Init (uint32_t AudioFreq)</code></td>
<td>Initializes the Audio Codec audio interface (SAI).</td>
</tr>
<tr>
<td><code>static void SAIx_In_DeInit (void)</code></td>
<td>Deinitializes the output Audio Codec audio interface (SAI).</td>
</tr>
<tr>
<td><code>static void DFSDMx_ChannelMspInit (void)</code></td>
<td>Initialize the DFSDM channel MSP.</td>
</tr>
<tr>
<td><code>static void DFSDMx_FilterMspInit (void)</code></td>
<td>Initialize the DFSDM filter MSP.</td>
</tr>
<tr>
<td><code>static void DFSDMx_ChannelMspDeInit (void)</code></td>
<td>DeInitialize the DFSDM channel MSP.</td>
</tr>
<tr>
<td><code>static void DFSDMx_FilterMspDeInit (void)</code></td>
<td>DeInitialize the DFSDM filter MSP.</td>
</tr>
<tr>
<td><code>static uint8_t DFSDMx_Init (uint32_t AudioFreq)</code></td>
<td>Initialize the Digital Filter for Sigma-Delta Modulators interface (DFSDM).</td>
</tr>
<tr>
<td><code>static uint8_t DFSDMx_DeInit (void)</code></td>
<td>De-initialize the Digital Filter for Sigma-Delta Modulators interface (DFSDM).</td>
</tr>
<tr>
<td><code>uint8_t BSP_AUDIO_OUT_Init (uint16_t OutputDevice, uint8_t Volume, uint32_t AudioFreq)</code></td>
<td>Initializes the Digital Filter for Sigma-Delta Modulators interface (DFSDM).</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>uint8_t BSP_AUDIO_OUT_Play(uint16_t *pBuffer, uint32_t Size)</code></td>
<td>Starts playing audio stream from a data buffer for a determined size.</td>
</tr>
<tr>
<td><code>void BSP_AUDIO_OUT_ChangeBuffer(uint16_t *pData, uint16_t Size)</code></td>
<td>Sends n-Bytes on the SAI interface.</td>
</tr>
<tr>
<td><code>uint8_t BSP_AUDIO_OUT_Pause(void)</code></td>
<td>This function Pauses the audio file stream.</td>
</tr>
<tr>
<td><code>uint8_t BSP_AUDIO_OUT_Resume(void)</code></td>
<td>Resumes the audio file stream.</td>
</tr>
<tr>
<td><code>uint8_t BSP_AUDIO_OUT_Stop(uint32_t Option)</code></td>
<td>Stops audio playing and Power down the Audio Codec.</td>
</tr>
<tr>
<td><code>uint8_t BSP_AUDIO_OUT_SetVolume(uint8_t Volume)</code></td>
<td>Controls the current audio volume level.</td>
</tr>
<tr>
<td><code>uint8_t BSP_AUDIO_OUT_SetMute(uint32_t Cmd)</code></td>
<td>Enables or disables the MUTE mode by software.</td>
</tr>
<tr>
<td><code>uint8_t BSP_AUDIO_OUT_SetOutputMode(uint8_t Output)</code></td>
<td>Switch dynamically (while audio file is played) the output target (speaker or headphone).</td>
</tr>
<tr>
<td><code>void BSP_AUDIO_OUT_SetFrequency(uint32_t AudioFreq)</code></td>
<td>Updates the audio frequency.</td>
</tr>
<tr>
<td><code>void BSP_AUDIO_OUT_SetAudioFrameSlot(uint32_t AudioFrameSlot)</code></td>
<td>Updates the Audio frame slot configuration.</td>
</tr>
<tr>
<td><code>void BSP_AUDIO_OUT_DeInit(void)</code></td>
<td>De-initializes the audio out peripheral.</td>
</tr>
<tr>
<td><code>void HAL_SAI_TxCpltCallback(SAI_HandleTypeDef *hsai)</code></td>
<td>Tx Transfer completed callbacks.</td>
</tr>
<tr>
<td><code>void HAL_SAI_TxHalfCpltCallback(SAI_HandleTypeDef *hsai)</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>HAL_SAI_ErrorCallback</code></td>
<td>SAI error callbacks.</td>
</tr>
<tr>
<td><code>BSP_AUDIO_OUT_TransferComplete_CallBack</code></td>
<td>Manages the DMA full Transfer complete event.</td>
</tr>
<tr>
<td><code>BSP_AUDIO_OUT_HalfTransfer_CallBack</code></td>
<td>Manages the DMA Half Transfer complete event.</td>
</tr>
<tr>
<td><code>BSP_AUDIO_OUT_Error_CallBack</code></td>
<td>Manages the DMA FIFO error event.</td>
</tr>
<tr>
<td><code>BSP_AUDIO_OUT_MspInit</code></td>
<td>Initializes BSP_AUDIO_OUT MSP.</td>
</tr>
<tr>
<td><code>BSP_AUDIO_OUT_MspDeInit</code></td>
<td>Deinitializes SAI MSP.</td>
</tr>
<tr>
<td><code>BSP_AUDIO_OUT_ClockConfig</code></td>
<td>Clock Config.</td>
</tr>
<tr>
<td><code>BSP_AUDIO_IN_Init</code></td>
<td>Initialize wave recording.</td>
</tr>
<tr>
<td><code>BSP_AUDIO_IN_InitEx</code></td>
<td>Initialize wave recording.</td>
</tr>
<tr>
<td><code>BSP_AUDIO_IN_AllocScratch</code></td>
<td>Allocate channel buffer scratch.</td>
</tr>
<tr>
<td><code>BSP_AUDIO_IN_GetChannelNumber</code></td>
<td>Return audio in channel number.</td>
</tr>
<tr>
<td><code>BSP_AUDIO_IN_Record</code></td>
<td>Start audio recording.</td>
</tr>
<tr>
<td><code>BSP_AUDIO_IN_Stop</code></td>
<td>Stop audio recording.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>uint8_t BSP_AUDIO_IN_Pause (void)</td>
<td>Pause the audio file stream.</td>
</tr>
<tr>
<td>uint8_t BSP_AUDIO_IN_Resume (void)</td>
<td>Resume the audio file stream.</td>
</tr>
<tr>
<td>void BSP_AUDIO_IN_DelInit (void)</td>
<td>Deinit the audio IN peripherals.</td>
</tr>
<tr>
<td>void HAL_DFSDM_FilterRegConvCpltCallback (DFSDM_Filter_HandleTypeDef *hdfsdm_filter)</td>
<td>Regular conversion complete callback.</td>
</tr>
<tr>
<td>void HAL_DFSDM_FilterRegConvHalfCpltCallback (DFSDM_Filter_HandleTypeDef *hdfsdm_filter)</td>
<td>Half regular conversion complete callback.</td>
</tr>
<tr>
<td>void HAL_SAI_RxHalfCpltCallback (SAI_HandleTypeDef *hsai)</td>
<td>Half reception complete callback.</td>
</tr>
<tr>
<td>void HAL_SAI_RxCpltCallback (SAI_HandleTypeDef *hsai)</td>
<td>Reception complete callback.</td>
</tr>
<tr>
<td>__weak void BSP_AUDIO_IN_TransferComplete_CallBack (void)</td>
<td>User callback when record buffer is filled.</td>
</tr>
<tr>
<td>__weak void BSP_AUDIO_IN_HalfTransfer_CallBack (void)</td>
<td>Manages the DMA Half Transfer complete event.</td>
</tr>
<tr>
<td>__weak void BSP_AUDIO_IN_Error_CallBack (void)</td>
<td>Audio IN Error callback function.</td>
</tr>
<tr>
<td>__weak void BSP_AUDIO_IN_MspInit (void)</td>
<td>Initialize BSP_AUDIO_IN MSP.</td>
</tr>
<tr>
<td>__weak void BSP_AUDIO_IN_MspDeInit (void)</td>
<td>DeInitialize BSP_AUDIO_IN MSP.</td>
</tr>
<tr>
<td>__weak void BSP_AUDIO_IN_ClockConfig</td>
<td>Clock Config.</td>
</tr>
</tbody>
</table>

The functions include:
- `BSP_AUDIO_IN_Pause`: Pauses the audio file stream.
- `BSP_AUDIO_IN_Resume`: Resumes the audio file stream.
- `BSP_AUDIO_IN_DelInit`: Deinitializes the audio IN peripherals.
- `HAL_DFSDM_FilterRegConvCpltCallback`: Handles regular conversion completion.
- `HAL_DFSDM_FilterRegConvHalfCpltCallback`: Handles half regular conversion completion.
- `HAL_SAI_RxHalfCpltCallback`: Handles half reception completion.
- `HAL_SAI_RxCpltCallback`: Handles reception completion.
- `BSP_AUDIO_IN_TransferComplete_CallBack`: Calls the user callback when the record buffer is filled.
- `BSPAUDIO_IN_HalfTransfer_CallBack`: Manages the DMA half transfer completion.
- `BSP_AUDIO_IN_Error_CallBack`: Handles audio IN error callbacks.
- `BSP_AUDIO_IN_MspInit`: Initializes the audio IN MSP.
- `BSP_AUDIO_IN_MspDeInit`: Deinitializes the audio IN MSP.
- `BSP_AUDIO_IN_ClockConfig`: Configures the audio IN clock.
### Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUDIO_DrvTypeDef *</td>
<td>audio_drv</td>
</tr>
<tr>
<td>SAI_HandleTypeDef</td>
<td>haudio_out_sai</td>
</tr>
<tr>
<td>SAI_HandleTypeDef</td>
<td>haudio_in_sai</td>
</tr>
<tr>
<td>AUDIOIN_TypeDef</td>
<td>hAudioIn</td>
</tr>
<tr>
<td>DFSDM_Channel_HandleTypeDef</td>
<td>hAudioInTopLeftChannel</td>
</tr>
<tr>
<td>DFSDM_Channel_HandleTypeDef</td>
<td>hAudioInTopRightChannel</td>
</tr>
<tr>
<td>DFSDM_Filter_HandleTypeDef</td>
<td>hAudioInTopLeftFilter</td>
</tr>
<tr>
<td>DFSDM_Filter_HandleTypeDef</td>
<td>hAudioInTopRightFilter</td>
</tr>
<tr>
<td>DMA_HandleTypeDef</td>
<td>hDmaTopLeft</td>
</tr>
<tr>
<td>DMA_HandleTypeDef</td>
<td>hDmaTopRight</td>
</tr>
<tr>
<td>DFSDM_Channel_HandleTypeDef</td>
<td>hAudioInBottomLeftChannel</td>
</tr>
<tr>
<td>DFSDM_Channel_HandleTypeDef</td>
<td>hAudioInBottomRightChannel</td>
</tr>
<tr>
<td>DFSDM_Filter_HandleTypeDef</td>
<td>hAudioInBottomLeftFilter</td>
</tr>
<tr>
<td>DFSDM_Filter_HandleTypeDef</td>
<td>hAudioInBottomRightFilter</td>
</tr>
<tr>
<td>DMA_HandleTypeDef</td>
<td>hDmaBottomLeft</td>
</tr>
<tr>
<td>DMA_HandleTypeDef</td>
<td>hDmaBottomRight</td>
</tr>
<tr>
<td>static int32_t *</td>
<td>pScratchBuff [2]</td>
</tr>
<tr>
<td>static __IO int32_t</td>
<td>ScratchSize</td>
</tr>
<tr>
<td>static uint8_t</td>
<td>AudioIn_ChannelNumber = DEFAULT_AUDIO_IN_CHANNEL</td>
</tr>
<tr>
<td>static uint16_t</td>
<td>AudioIn_Device = INPUT_DEVICE_DIGITAL_MIC</td>
</tr>
<tr>
<td>static uint32_t</td>
<td>DmaTopLeftRecHalfCplt = 0</td>
</tr>
<tr>
<td>static uint32_t</td>
<td>DmaTopLeftRecCplt = 0</td>
</tr>
<tr>
<td>static uint32_t</td>
<td>DmaTopRightRecHalfCplt = 0</td>
</tr>
<tr>
<td>static uint32_t</td>
<td>DmaTopRightRecCplt = 0</td>
</tr>
<tr>
<td>static uint32_t</td>
<td>DmaBottomLeftRecHalfCplt = 0</td>
</tr>
<tr>
<td>static uint32_t</td>
<td>DmaBottomLeftRecCplt = 0</td>
</tr>
<tr>
<td>static uint32_t</td>
<td>DmaBottomRightRecHalfCplt = 0</td>
</tr>
<tr>
<td>static uint32_t</td>
<td>DmaBottomRightRecCplt = 0</td>
</tr>
<tr>
<td>Static Type</td>
<td>Variable Name</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>__IO uint32_t</td>
<td>AppBuffTrigger</td>
</tr>
<tr>
<td>__IO uint32_t</td>
<td>AppBuffHalf</td>
</tr>
</tbody>
</table>
Detailed Description

This file provides the Audio driver for the STM32F769I-DISCOVERY board.

Author:
MCD Application Team

Version:
V2.0.0

Date:
30-December-2016

How To use this driver:
-----------------------
+ This driver supports STM32F7xx devices on STM32F769I-DISCOVERY (MB1225) Evaluation boards.
+ Call the function BSP_AUDIO_OUT_Init(
   OutputDevice:
      physical output mode (OUTPUT_DEVICE_SPEAKER,
      OUTPUT_DEVICE_HEADPHONE or OUTPUT_DEVICE_BOTH)
   Volume
      : Initial volume to be set (0 is min (mute), 100 is max (100%))
   AudioFreq
      : Audio frequency in Hz (8000, 16000, 22500, 32000...)
   this parameter is relative to the audio file/stream type.
)
This function configures all the hardwar
e required for the audio application (codec, I2C, SAI, GPIOs, DMA and interrupt if needed). This function returns AUDIO_OK if configuration is OK.

If the returned value is different from AUDIO_OK or the function is stuck then the communication with the codec has failed (try to un-plug the power or reset device in this case).

- OUTPUT_DEVICE_SPEAKER : only speaker will be set as output for the audio stream.
- OUTPUT DEVICE_HEADPHONE: only headphones will be set as output for the audio stream.
- OUTPUT DEVICE BOTH : both Speaker and Headphone are used as outputs for the audio stream at the same time.

Note. On STM32F769I-DISCOVERY SAI_DMA is configured in CIRCULAR mode. Due to this the application does NOT need to call BSP_AUDIO_OUT_ChangeBuffer() to assure streaming.

+ Call the function BSP_AUDIO_OUT_Play(pBuffer: pointer to the audio data file address, Size : size of the buffer to be sent in Bytes)

  to start playing (for the first time) from the audio file/stream.
+ Call the function BSP_AUDIO_OUT_Pause() to pause playing
+ Call the function BSP_AUDIO_OUT_Resume() to resume playing.

Note. After calling BSP_AUDIO_OUT_Pause
() function for pause, only BSP_AUDIO_OUT_Resume() should be called for resume (it is not allowed to call BSP_AUDIO_OUT_Play() in this case).

Note. This function should be called only when the audio file is played or paused (not stopped).

+ For each mode, you may need to implement the relative callback functions into your code.

The Callback functions are named BSP_AUDIO_OUT_XXX_CallBack() and only their prototypes are declared in the stm32f769i_discovery_audio.h file. (refer to the example for more details on the callbacks implementations)

+ To Stop playing, to modify the volume level, the frequency, the audio frame slot, the device output mode the mute or the stop, use the functions: BSP_AUDIO_OUT_SetVolume(), AUDIO_OUT_SetFrequency(), BSP_AUDIO_OUT_SetAudioFrameSlot(), BSP_AUDIO_OUT_SetOutputMode(), BSP_AUDIO_OUT_SetMute() and BSP_AUDIO_OUT_Stop().

+ Call the function BSP_AUDIO_IN_Init(

  AudioFreq: Audio frequency in Hz (8000, 16000, 22500, 32000...)

  this parameter is relative to the audio file/stream type.

  BitRes: Bit resolution fixed to 16bit

  ChnlNbr: N
number of channel to be configured for the DFSDM peripheral

This function configures all the hardware required for the audio in application (DFSDM filters and channels,
Clock source for DFSDM peripheral, GPIOs, DMA and interrupt if needed).
This function returns AUDIO_OK if configuration is OK. If the returned value is different from AUDIO_OK then
the configuration should be wrong.
Note: On STM32F769I-DISCOVERY, four DFSDM Channel/Filters are configured and their DMA streams are configured
in CIRCULAR mode.

+ Call the function BSP_AUDIO_IN_AllocScratch(
pScratch: pointer to scratch tables
size:
size of scratch buffer)
This function must be called before BSP_AUDIO_IN_RECORD() to allocate buffer scratch for each DFSDM channel
and its size.
Note: These buffers scratch are used as intermediate buffers to collect data within final record buffer.
size is the total size of the four buffers scratch; If size is 512 then the size of each is 128.
This function must be called after BSP_AUDIO_IN_Init()

+ Call the function BSP_AUDIO_IN_RECORD(
pBuf: pointer to the recorded audio data file address
Size: size of buffer to be written in Bytes

to start recording from microphones.

+ Call the function BSP_AUDIO_IN_Pause() to pause recording
  + Call the function BSP_AUDIO_IN_Resume() to resume recording.

  Note. After calling BSP_AUDIO_IN_Pause() function for pause, only BSP_AUDIO_IN_Resume() should be called for resume (it is not allowed to call BSP_AUDIO_IN_RECORD() in this case).

+ Call the function BSP_AUDIO_IN_Stop() to stop recording
  + For each mode, you may need to implement the relative callback functions into your code.

  The Callback functions are named BSP_AUDIO_IN_XXX_CallBack() and only their prototypes are declared in the stm32f769i_discovery_audio.h file. (refer to the example for more details on the callbacks implementations)

Driver architecture:
---------------------
+ This driver provides the High Audio Layer: consists of the function API exported in the stm32f769i_discovery_audio.h file (BSP_AUDIO_OUT_Init(), BSP_AUDIO_OUT_Play() ...)
  + This driver provides also the Media Access Layer (MAL): which consists of functions allowing to access the media containing/ providing the audio file/stream. These fu
nctions are also included as local functions into the `stm32f769i_discovery_audio.c` file (DFSDMx_Init(), DFSDMx_DeInit(), SAIx_Init() and SAIx_DeInit())

Known Limitations:
------------------
1- If the TDM Format used to play in parallel 2 audio Stream (the first Stream is configured in codec SLOT0 and second Stream in SLOT1) the Pause/Resume, volume and mute feature will control the both streams.

2- Parsing of audio file is not implemented (in order to determine audio file properties: Mono/Stereo, Data size, File size, Audio Frequency, Audio Data header size ...). The configuration is fixed for the given audio file.

3- Supports only Stereo audio streaming.

4- Supports only 16-bits audio data size.

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

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

Definition in file stm32f769i_discovery_audio.c.
This file contains the common defines and functions prototypes for the `stm32f769i_discovery_audio.c` driver. More...

```
#include "../Components/wm8994/wm8994.h"
#include "stm32f769i_discovery.h"
#include <stdlib.h>
```

Go to the source code of this file.
Defines

#define BSP_AUDIO_FREQUENCY_96K SAI_AUDIO_FREQUENCY_96K
#define BSP_AUDIO_FREQUENCY_48K SAI_AUDIO_FREQUENCY_48K
#define BSP_AUDIO_FREQUENCY_44K SAI_AUDIO_FREQUENCY_44K
#define BSP_AUDIO_FREQUENCY_32K SAI_AUDIO_FREQUENCY_32K
#define BSP_AUDIO_FREQUENCY_22K SAI_AUDIO_FREQUENCY_22K
#define BSP_AUDIO_FREQUENCY_16K SAI_AUDIO_FREQUENCY_16K
#define BSP_AUDIO_FREQUENCY_11K SAI_AUDIO_FREQUENCY_11K
#define BSP_AUDIO_FREQUENCY_8K SAI_AUDIO_FREQUENCY_8K
#define CODEC_AUDIOFRAME_SLOT_0123 SAI_SLOTACTIVE_0 | SAI_SLOTACTIVE_2 | SAI_SLOTACTIVE_3
 CODEC_AudioFrame_SLOT_TDMMode In W8994 codec the A
 Mode TDM format: +------------------|------------------|--------------------|-------------------+	|
 CODEC_SLOT0 Left | CODEC_SLOT1 Left | CODEC_SLOT0 Right |	+----
--------------------------------------------------------------------------+
#define CODEC_AUDIOFRAME_SLOT_02 SAI_SLOTACTIVE_0 | SAI_SLOTACTIVE_2
#define CODEC_AUDIOFRAME_SLOT_13 SAI_SLOTACTIVE_1 | SAI_SLOTACTIVE_3
#define AUDIO_OUT_SAIx SAI1_Block_A
#define AUDIO_OUT_SAIx_CLK_ENABLE() __HAL_RCC_SAI1_CLK_ENABLE()
#define AUDIO_OUT_SAIx_CLK_DISABLE() __HAL_RCC_SAI1_CLK_DISABLE()
#define AUDIO_OUT_SAIx_AF GPIO_AF6_SAI1
#define AUDIO_OUT_SAIx_MCLK_ENABLE() __HAL_RCC_GPIOG_CLK_ENABLE()
#define AUDIO_OUT_SAIx_MCLK_GPIO_PORT GPIOG
#define AUDIO_OUT_SAIx_MCLK_PIN GPIO_PIN_7
#define AUDIO_OUT_SAIx_SD_FS_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define AUDIO_OUT_SAIx_SD_FS_SCK_GPIO_PORT GPIOE
#define AUDIO_OUT_SAIx_SD_PIN GPIO_PIN_6
#define AUDIO_OUT_SAIx_DMAx_CLK_ENABLE() __HAL_RCC_DMA2_CLK_ENABLE()
#define AUDIO_OUT_SAIx_DMAx_STREAM DMA2_Stream1
#define AUDIO_OUT_SAIx_DMAx_CHANNEL DMA_CHANNEL_0
#define AUDIO_OUT_SAIx_DMAx_IRQ DMA2_Stream1_IRQHandler
#define AUDIO_OUT_SAIx_DMAx_PERIPH_DATA_SIZE DMA_PDATAALIGN_HALFWORD
#define AUDIO_OUT_SAIx_DMAx_MEM_DATA_SIZE DMA_MDATAALIGN_HALFWORD
#define DMA_MAX_SZE 0xFFFF
#define AUDIO_OUT_SAIx_DMAx_IRQHandler DMA2_Stream1_IRQHandler
#define AUDIO_OUT_IRQ_PREPRIO ((uint32_t)0x0E)
#define AUDIO_IN_SAIx SAI1_Block_B
#define AUDIO_IN_SAIx_CLK_ENABLE () __HAL_RCC_SAI1_CLK_ENABLE()
#define AUDIO_IN_SAIx_CLK_DISABLE () __HAL_RCC_SAI1_CLK_DISABLE()
#define AUDIO_IN_SAIx_AF GPIO_AF6_SAI1
#define AUDIO_IN_SAIx_SD_ENABLE () __HAL_RCC_GPIOE_CLK_ENABLE()
#define AUDIO_IN_SAIx_SD_GPIO_PORT GPIOE
#define AUDIO_IN_SAIx_SD_PIN GPIO_PIN_3
#define AUDIO_IN_SAIx_DMAx_CLK_ENABLE () __HAL_RCC_DMA2_CLK_ENABLE()
#define AUDIO_IN_SAIx_DMAx_STREAM DMA2_Stream4
#define AUDIO_IN_SAIx_DMAx_CHANNEL DMA_CHANNEL_1
#define AUDIO_IN_SAIx_DMAx_IRQ DMA2_Stream4_IRQHandler
#define AUDIO_IN_SAIx_DMAx_PERIPH_DATA_SIZE DMA_PDATAALIGN_HALFWORD
#define AUDIO_IN_SAIx_DMAx_MEM_DATA_SIZE DMA_MDATAALIGN_HALFWORD
#define AUDIO_IN_INT_GPIO_ENABLE () __HAL_RCC_GPIOJ_CLK_ENABLE()
#define AUDIO_IN_INT_GPIO_PORT GPIOJ
#define AUDIO_IN_INT_GPIO_PIN GPIO_PIN_12
#define AUDIO_IN_INT_IRQ EXTI15_10_IRQn
#define AUDIO_DFSDMx_TOP_RIGHT_CHANNEL DFSDM_CHANNEL_0
#define AUDIO_DFSDMx_TOP_LEFT_CHANNEL DFSDM_CHANNEL_1
#define AUDIO_DFSDMx_BUTTOM_RIGHT_CHANNEL DFSDM_CHANNEL_4
#define AUDIO_DFSDMx_BUTTOM_LEFT_CHANNEL DFSDM_CHANNEL_5
#define AUDIO_DFSDMx_TOP_LEFT_FILTER DFSDM1_Filter0
#define AUDIO_DFSDMx_TOP_RIGHT_FILTER DFSDM1_Filter1
#define AUDIO_DFSDMx_BUTTOM_LEFT_FILTER DFSDM1_Filter2
#define AUDIO_DFSDMx_BUTTOM_RIGHT_FILTER DFSDM1_Filter3
#define AUDIO_DFSDMx_CLK_ENABLE () __HAL_RCC_DFSDM1_CLK_ENABLE()
#define AUDIO_DFSDMx_CKOUT_PIN GPIO_PIN_3
#define AUDIO_DFSDMx_CKOUT_DMIC_GPIO_PORT GPIOD
#define AUDIO_DFSDMx_CKOUT_DMIC_GPIO_CLK_ENABLE()
#define AUDIO_DFSDMx_DMIC_DATIN1_PIN GPIO_PIN_3
#define AUDIO_DFSDMx_DMIC_DATIN5_PIN GPIO_PIN_11
#define AUDIO_DFSDMx_DMIC_DATIN_GPIO_PORT GPIOC
#define AUDIO_DFSDMx_DMIC_DATIN_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define AUDIO_DFSDMx_DMIC_DATIN_AF GPIO_AF3_DFSDM1
#define AUDIO_DFSDMx_CKOUT_AF GPIO_AF3_DFSDM1
#define AUDIO_DFSDMx_DMAx_CLK_ENABLE() __HAL_RCC_DMA2_CLK_ENABLE()
#define AUDIO_DFSDMx_DMAx_CHANNEL DMA_CHANNEL_8
#define AUDIO_DFSDMx_DMAx_PERIPH_DATA_SIZE DMA_PDATAALIGN_WORD
#define AUDIO_DFSDMx_DMAx_MEM_DATA_SIZE DMA_MDATAALIGN_WORD
#define AUDIO_DFSDMx_DMAx_TOP_LEFT_STREAM DMA2_Stream0
#define AUDIO_DFSDMx_DMAx_TOP_LEFT_IRQ DMA2_Stream0_IRQn
#define AUDIO_DFSDMx_DMAx_TOP_LEFT_IRQHandler DMA2_Stream0_IRQHandler
#define AUDIO_DFSDMx_DMAx_TOP_RIGHT_STREAM DMA2_Stream5
#define AUDIO_DFSDMx_DMAx_TOP_RIGHT_IRQ DMA2_Stream5_IRQn
#define AUDIO_DFSDMx_DMAx_TOP_RIGHT_IRQHandler DMA2_Stream5_IRQHandler
#define AUDIO_DFSDMx_DMAx_BUTTOM_LEFT_STREAM DMA2_Stream6
#define AUDIO_DFSDMx_DMAx_BUTTOM_LEFT_IRQ DMA2_Stream6_IRQn
#define AUDIO_DFSDMx_DMAx_BUTTOM_LEFT_IRQHandler DMA2_Stream6_IRQHandler
#define AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_STREAM DMA2_Stream7
#define AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_IRQ DMA2_Stream7_IRQn
#define AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_IRQHandler DMA2_Stream7_IRQHandler
#define AUDIO_IN_IRQ_PREPRIO ((uint32_t)0x0F)
#define AUDIODATA_SIZE 2 /* 16-bits audio data size */
#define AUDIO_OK ((uint8_t)0)
#define AUDIO_ERROR ((uint8_t)1)
#define AUDIO_TIMEOUT ((uint8_t)2)
#define DEFAULT_AUDIO_IN_FREQ BSP_AUDIO_FREQUENCY_16K
#define DEFAULT_AUDIO_IN_BIT_RESOLUTION ((uint8_t)16)
#define DEFAULT_AUDIO_IN_CHANNEL_NBR ((uint8_t)2)
#define DEFAULT_AUDIO_IN_VOLUME ((uint16_t)64)
#define OUTPUT_DEVICE_HEADPHONE1 OUTPUT_DEVICE_HEADPHONE	/* Headphone2 is connected */
#define OUTPUT_DEVICE_HEADPHONE2 OUTPUT_DEVICE_SPEAKER
to Speaker output of the wm8994 */

#define INPUT_DEVICE_DIGITAL_MIC   ((uint16_t)0)
#define INPUT_DEVICE_ANALOG_MIC     INPUT_DEVICE_INPUT_LINE_1
#define DMA_MAX(x)  (((x) <= DMA_MAX_SZE)? (x):DMA_MAX_SZE)
Functions

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Function Name</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td><strong>BSP_AUDIO_OUT_Init</strong> (uint16_t OutputDevice, uint8_t Volume, uint32_t AudioFreq)</td>
<td></td>
<td>Configures the audio peripherals.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_AUDIO_OUT_DeInit</strong> (void)</td>
<td></td>
<td>De-initializes the audio out peripheral.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_AUDIO_OUT_Play</strong> (uint16_t *pBuffer, uint32_t Size)</td>
<td></td>
<td>Starts playing audio stream from a data buffer for a determined size.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_AUDIO_OUT_ChangeBuffer</strong> (uint16_t *pData, uint16_t Size)</td>
<td></td>
<td>Sends n-Bytes on the SAI interface.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_AUDIO_OUT_Pause</strong> (void)</td>
<td></td>
<td>This function Pauses the audio file stream.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_AUDIO_OUT_Resume</strong> (void)</td>
<td></td>
<td>Resumes the audio file stream.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_AUDIO_OUT_Stop</strong> (uint32_t Option)</td>
<td></td>
<td>Stops audio playing and Power down the Audio Codec.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_AUDIO_OUT_SetVolume</strong> (uint8_t Volume)</td>
<td></td>
<td>Controls the current audio volume level.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_AUDIO_OUT_SetFrequency</strong> (uint32_t AudioFreq)</td>
<td></td>
<td>Updates the audio frequency.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_AUDIO_OUT_SetAudioFrameSlot</strong> (uint32_t AudioFrameSlot)</td>
<td></td>
<td>Updates the Audio frame slot configuration.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_AUDIO_OUT_SetMute</strong> (uint32_t Cmd)</td>
<td></td>
<td>Enables or disables the MUTE mode by software.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_AUDIO_OUT_SetOutputMode</strong> (uint8_t Output)</td>
<td></td>
<td>Switch dynamically (while audio file is played) the output target (speaker or headphone).</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_AUDIO_OUT_TransferComplete_CallBack</strong> (void)</td>
<td></td>
<td>Manages the DMA full Transfer complete event.</td>
</tr>
<tr>
<td>Function Type</td>
<td>Function Name</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>BSP_AUDIO_OUT_HalfTransfer_CallBack (void)</td>
<td>Manages the DMA Half Transfer complete event.</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>BSP_AUDIO_OUT_Error_CallBack (void)</td>
<td>Manages the DMA FIFO error event.</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>BSP_AUDIO_OUT_ClockConfig (SAI_HandleTypeDef *hsai, uint32_t AudioFreq, void *Params)</td>
<td>Clock Config.</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>BSP_AUDIO_OUT_MspInit (SAI_HandleTypeDef *hsai, void *Params)</td>
<td>Initializes BSP_AUDIO_OUT MSP.</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>BSP_AUDIO_OUT_MspDeInit (SAI_HandleTypeDef *hsai, void *Params)</td>
<td>Deinitializes SAI MSP.</td>
<td></td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_AUDIO_IN_Init (uint32_t AudioFreq, uint32_t BitRes, uint32_t ChnlNbr)</td>
<td>Initialize wave recording.</td>
<td></td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_AUDIO_IN_InitEx (uint16_t InputDevice, uint32_t AudioFreq, uint32_t BitRes, uint32_t ChnlNbr)</td>
<td>Initialize wave recording.</td>
<td></td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_AUDIO_IN_AllocScratch (int32_t *pScratch, uint32_t size)</td>
<td>Allocate channel buffer scratch.</td>
<td></td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_AUDIO_IN_GetChannelNumber (void)</td>
<td>Return audio in channel number.</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>BSP_AUDIO_IN_DeInit (void)</td>
<td>Deinit the audio IN peripherals.</td>
<td></td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_AUDIO_IN_Record (uint16_t *pData, uint32_t Size)</td>
<td>Start audio recording.</td>
<td></td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_AUDIO_IN_Stop (void)</td>
<td>Stop audio recording.</td>
<td></td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_AUDIO_IN_Pause (void)</td>
<td>Pause the audio file stream.</td>
<td></td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_AUDIO_IN_Resume (void)</td>
<td>Resume the audio file stream.</td>
<td></td>
</tr>
<tr>
<td>Function Name</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>--------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BSP_AUDIO_IN_TransferComplete_CallBack</strong> (void)</td>
<td>User callback when record buffer is filled.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BSP_AUDIO_IN_HalfTransfer_CallBack</strong> (void)</td>
<td>Manages the DMA Half Transfer complete event.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BSP_AUDIO_IN_Error_CallBack</strong> (void)</td>
<td>Audio IN Error callback function.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BSP_AUDIO_IN_ClockConfig</strong></td>
<td>Clock Config.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BSP_AUDIO_IN_MspInit</strong> (void)</td>
<td>Initialize BSP_AUDIO_IN MSP.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BSP_AUDIO_IN_MspDeInit</strong> (void)</td>
<td>DeInitialize BSP_AUDIO_IN MSP.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Detailed Description

This file contains the common defines and functions prototypes for the *stm32f769i_discovery_audio.c* driver.

**Author:**
MCD Application Team

**Version:**
V2.0.0

**Date:**
30-December-2016

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

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

Definition in file stm32f769i_discovery_audio.h.
stm32f769i_discovery_eeprom.c File Reference

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

#include "stm32f769i_discovery_eeprom.h"

Go to the source code of this file.
### Functions

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint32_t</td>
<td><code>BSP_EEPROM_Init</code> (void)</td>
<td>Initializes peripherals used by the I2C EEPROM driver.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><code>BSP_EEPROM_DelInit</code> (void)</td>
<td>DeInitializes the EEPROM.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><code>BSP_EEPROM_ReadBuffer</code> (uint8_t *pBuffer, uint16_t ReadAddr, uint16_t *NumByteToRead)</td>
<td>Reads a block of data from the EEPROM.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><code>BSP_EEPROM_WritePage</code> (uint8_t *pBuffer, uint16_t WriteAddr, uint8_t *NumByteToWrite)</td>
<td>Writes more than one byte to the EEPROM with a single WRITE cycle.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><code>BSP_EEPROM_WriteBuffer</code> (uint8_t *pBuffer, uint16_t WriteAddr, uint16_t NumByteToWrite)</td>
<td>Writes buffer of data to the I2C EEPROM.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><code>BSP_EEPROM_WaitEepromStandbyState</code> (void)</td>
<td>Wait for EEPROM Standby state.</td>
</tr>
<tr>
<td>__weak void</td>
<td><code>BSP_EEPROM_TIMEOUT_UserCallback</code> (void)</td>
<td>Basic management of the timeout situation.</td>
</tr>
</tbody>
</table>
### Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__IO uint16_t</td>
<td>EEPROMAddress</td>
<td>= 0</td>
</tr>
<tr>
<td>__IO uint16_t</td>
<td>EEPROMDataRead</td>
<td></td>
</tr>
<tr>
<td>__IO uint8_t</td>
<td>EEPROMDataWrite</td>
<td></td>
</tr>
</tbody>
</table>
Detailed Description

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

Author:
MCD Application Team

Version:
V2.0.0

Date:
30-December-2016

To be able to use this driver, the switch EE_M24LR64 must be defined in your toolchain compiler preprocessor.

================================================================================================

Notes:
- The I2C EEPROM memory (M24LR64) is available on separate daughter board ANT7-M24LR-A, which is not provided with the STM32F769I_DISCOVERY board.
To use this driver you have to connect the ANT7-M24LR-A to CN2 connector of STM32F769I_DISCOVERY board.

================================================================================================

It implements a high level communication layer for read and write from/to this memory. The needed ST
M32F7xx hardware resources (I2C and GPIO) are defined in stm32f769i_discovery.h file, and the initialization is performed in EEPROM_IO_Init() function declared in stm32f769i_discovery.c file.

You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and EEPROM_IO_Init() function.

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

@note Regarding BSP_EEPROM_WritePage(), it is an optimized function to perform small write (less than 1 page) BUT the number of bytes (combined to write start address) must not cross the EEPROM page boundary. This function can only writes into the boundaries of an EEPROM page.

This function doesn't check on boundaries condition (in this driver the function BSP_EEPROM_WriteBuffer() which calls BSP_EEPROM_WritePage() is responsible of checking on Page boundaries).
### Pin assignment for M24LR64 EEPROM

<table>
<thead>
<tr>
<th>STM32F7xx I2C Pins</th>
<th>Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0(GND)</td>
<td>1 (0V)</td>
</tr>
<tr>
<td>AC0</td>
<td>2</td>
</tr>
<tr>
<td>AC1</td>
<td>3</td>
</tr>
<tr>
<td>VSS</td>
<td>4 (0V)</td>
</tr>
<tr>
<td>SDA</td>
<td>5</td>
</tr>
<tr>
<td>SCL</td>
<td>6</td>
</tr>
<tr>
<td>E1(GND)</td>
<td>7 (0V)</td>
</tr>
<tr>
<td>VDD</td>
<td>8 (3.3V)</td>
</tr>
</tbody>
</table>

---

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

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

Definition in file stm32f769i_discovery_eeprom.c.
stm32f769i_discovery_eeprom.h File Reference

This file contains all the functions prototypes for the stm32f769i_discovery_eeprom.c firmware driver. More...

#include "stm32f769i_discovery.h"

Go to the source code of this file.
### Defines

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

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint32_t</td>
<td><strong>BSPEEPROMInit</strong>(void)</td>
<td>Initializes peripherals used by the I2C EEPROM driver.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSPEEPROMDeInit</strong>(void)</td>
<td>DeInitializes the EEPROM.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><strong>BSPEEPROMReadBuffer</strong>(uint8_t*pBuffer, uint16_t ReadAddr, uint16_t *NumByteToRead)</td>
<td>Reads a block of data from the EEPROM.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><strong>BSPEEPROMWritePage</strong>(uint8_t<em>pBuffer, uint16_t WriteAddr, uint8_t</em>NumByteToWrite)</td>
<td>Writes more than one byte to the EEPROM with a single WRITE cycle.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><strong>BSPEEPROMWriteBuffer</strong>(uint8_t*pBuffer, uint16_t WriteAddr, uint16_t NumByteToWrite)</td>
<td>Writes buffer of data to the I2C EEPROM.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><strong>BSPEEPROMWaitEepromStandbyState</strong>(void)</td>
<td>Wait for EEPROM Standby state.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSPEEPROMTIMEOUT_UserCallback</strong>(void)</td>
<td>Basic management of the timeout situation.</td>
</tr>
<tr>
<td>void</td>
<td><strong>EEPROM_IO_Init</strong>(void)</td>
<td>Initializes peripherals used by the I2C EEPROM driver.</td>
</tr>
<tr>
<td>HAL_StatusTypeDef</td>
<td><strong>EEPROM_IO_WriteData</strong>(uint16_t DevAddress, uint16_t MemAddress, uint8_t*pBuffer, uint32_t BufferSize)</td>
<td>Write data to I2C EEPROM driver in using DMA channel.</td>
</tr>
<tr>
<td>HAL_StatusTypeDef</td>
<td><strong>EEPROM_IO_ReadData</strong>(uint16_t DevAddress, uint16_t MemAddress, uint8_t*pBuffer, uint32_t BufferSize)</td>
<td></td>
</tr>
<tr>
<td>HAL_StatusTypeDef</td>
<td><strong>EEPROM_IO_IsDeviceReady</strong> (uint16_t DevAddress, uint32_t Trials)</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Checks if target device is ready for communication.</td>
<td></td>
</tr>
</tbody>
</table>

Read data from I2C EEPROM driver in using DMA channel.
Detailed Description

This file contains all the functions prototypes for the
stm32f769i_discovery_eeprom.c firmware driver.

Author:
MCD Application Team

Version:
V2.0.0

Date:
30-December-2016

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

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

Definition in file stm32f769i_discovery_eeprom.h.
This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM32F769I-DISCOVERY board. More...

```c
#include "stm32f769i_discovery_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**

<table>
<thead>
<tr>
<th>Define</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#define LCD_DSI_ID 0x11</td>
<td></td>
</tr>
<tr>
<td>#define LCD_DSI_ADDRESS TS_I2C_ADDRESS</td>
<td></td>
</tr>
<tr>
<td>#define LCD_DSI_ID_REG 0xA8</td>
<td></td>
</tr>
<tr>
<td>#define ABS(X) ((X) &gt; 0 ? (X) : -(X))</td>
<td></td>
</tr>
<tr>
<td>#define POLY_X(Z) ((int32_t)((Points + (Z))-&gt;X))</td>
<td></td>
</tr>
<tr>
<td>#define POLY_Y(Z) ((int32_t)((Points + (Z))-&gt;Y))</td>
<td></td>
</tr>
</tbody>
</table>
## Functions

<table>
<thead>
<tr>
<th>static void DrawChar (uint16_t Xpos, uint16_t Ypos, const uint8_t *c)</th>
<th>Draws a character on LCD.</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void FillTriangle (uint16_t x1, uint16_t x2, uint16_t x3, uint16_t y1, uint16_t y2, uint16_t y3)</td>
<td>Fills a triangle (between 3 points).</td>
</tr>
<tr>
<td>static void LL_FillBuffer (uint32_t LayerIndex, void *pDst, uint32_t xSize, uint32_t ySize, uint32_t OffLine, uint32_t ColorIndex)</td>
<td>Fills a buffer.</td>
</tr>
<tr>
<td>static void LL_ConvertLineToARGB8888 (void *pSrc, void *pDst, uint32_t xSize, uint32_t ColorMode)</td>
<td>Converts a line to an ARGB8888 pixel format.</td>
</tr>
<tr>
<td>static uint16_t LCD_IO_GetID (void)</td>
<td>Returns the ID of connected screen by checking the HDMI (adv7533 component) ID or LCD DSI (via TS ID) ID.</td>
</tr>
<tr>
<td>uint8_t BSP_LCD_Init (void)</td>
<td>Initializes the DSI LCD.</td>
</tr>
<tr>
<td>uint8_t BSP_LCD_InitEx (LCD_OrientationTypeDef orientation)</td>
<td>Initializes the DSI LCD.</td>
</tr>
<tr>
<td>void BSP_LCD_Reset (void)</td>
<td>BSP LCD Reset Hw reset the LCD DSI activating its XRES signal (active low for some time) and desactivating it later.</td>
</tr>
<tr>
<td>uint32_t BSP_LCD_GetXSize (void)</td>
<td>Gets the LCD X size.</td>
</tr>
<tr>
<td>uint32_t BSP_LCD_GetYSize (void)</td>
<td>Gets the LCD Y size.</td>
</tr>
<tr>
<td>void BSP_LCD_SetXSize (uint32_t imageWidthPixels)</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>void BSP_LCD_SetYSize</strong> (uint32_t imageHeightPixels)</td>
<td>Set the LCD Y size.</td>
</tr>
<tr>
<td><strong>void BSP_LCD_LayerDefaultInit</strong> (uint16_t LayerIndex, uint32_t FB_Address)</td>
<td>Initializes the LCD layers.</td>
</tr>
<tr>
<td><strong>void BSP_LCD_SelectLayer</strong> (uint32_t LayerIndex)</td>
<td>Selects the LCD Layer.</td>
</tr>
<tr>
<td><strong>void BSP_LCD_SetLayerVisible</strong> (uint32_t LayerIndex, FunctionalState State)</td>
<td>Sets an LCD Layer visible.</td>
</tr>
<tr>
<td><strong>void BSP_LCD_SetTransparency</strong> (uint32_t LayerIndex, uint8_t Transparency)</td>
<td>Configures the transparency.</td>
</tr>
<tr>
<td><strong>void BSP_LCD_SetLayerAddress</strong> (uint32_t LayerIndex, uint32_t Address)</td>
<td>Sets an LCD layer frame buffer address.</td>
</tr>
<tr>
<td><strong>void BSP_LCD_SetLayerWindow</strong> (uint16_t LayerIndex, uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height)</td>
<td>Sets display window.</td>
</tr>
<tr>
<td><strong>void BSP_LCD_SetColorKeying</strong> (uint32_t LayerIndex, uint32_t RGBValue)</td>
<td>Configures and sets the color keying.</td>
</tr>
<tr>
<td><strong>void BSP_LCD_ResetColorKeying</strong> (uint32_t LayerIndex)</td>
<td>Disables the color keying.</td>
</tr>
<tr>
<td><strong>void BSP_LCD_SetTextColor</strong> (uint32_t Color)</td>
<td>Sets the LCD text color.</td>
</tr>
<tr>
<td><strong>uint32_t BSP_LCD_GetTextColor</strong> (void)</td>
<td>Gets the LCD text color.</td>
</tr>
<tr>
<td><strong>void BSP_LCD_SetBackColor</strong> (uint32_t Color)</td>
<td>Sets the LCD background color.</td>
</tr>
<tr>
<td><strong>uint32_t BSP_LCD_GetBackColor</strong> (void)</td>
<td>Gets the LCD background color.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void BSP_LCD_SetFont (sFONT *fonts)</code></td>
<td>Sets the LCD text font.</td>
</tr>
<tr>
<td><code>sFONT * BSP_LCD_GetFont (void)</code></td>
<td>Gets the LCD text font.</td>
</tr>
<tr>
<td><code>uint32_t BSP_LCD_ReadPixel (uint16_t Xpos, uint16_t Ypos)</code></td>
<td>Reads an LCD pixel.</td>
</tr>
<tr>
<td><code>void BSP_LCD_Clear (uint32_t Color)</code></td>
<td>Clears the whole currently active layer of LTDC.</td>
</tr>
<tr>
<td><code>void BSP_LCD_ClearStringLine (uint32_t Line)</code></td>
<td>Clears the selected line in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_DisplayChar (uint16_t Xpos, uint16_t Ypos, uint8_t Ascii)</code></td>
<td>Displays one character in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_DisplayStringAt (uint16_t Xpos, uint16_t Ypos, uint8_t *Text, Text_AlignModeTypdef Mode)</code></td>
<td>Displays characters in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_DisplayStringAtLine (uint16_t Line, uint8_t *ptr)</code></td>
<td>Displays a maximum of 60 characters on the LCD.</td>
</tr>
<tr>
<td><code>void BSP_LCD_DrawHLine (uint16_t Xpos, uint16_t Ypos, uint16_t Length)</code></td>
<td>Draws an horizontal line in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_DrawVLine (uint16_t Xpos, uint16_t Ypos, uint16_t Length)</code></td>
<td>Draws a vertical line in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_DrawLine (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2)</code></td>
<td>Draws an uni-line (between two points) in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_DrawRect (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height)</code></td>
<td>Draws a rectangle in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_DrawCircle (uint16_t Xpos, uint16_t Ypos, uint16_t Radius)</code></td>
<td>Draws a circle in currently active layer.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void BSP_LCD_DrawPolygon (pPoint Points, uint16_t PointCount)</code></td>
<td>Draws a polygon (between many points) in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_DrawEllipse (int Xpos, int Ypos, int XRADIUS, int YRadius)</code></td>
<td>Draws an ellipse on LCD in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_DrawBitmap (uint32_t Xpos, uint32_t Ypos, uint8_t *pbmp)</code></td>
<td>Draws a bitmap picture loaded into the internal Flash (32 bpp) in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_FillRect (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height)</code></td>
<td>Draws a full rectangle in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_FillCircle (uint16_t Xpos, uint16_t Ypos, uint16_t Radius)</code></td>
<td>Draws a full circle in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_FillPolygon (pPoint Points, uint16_t PointCount)</code></td>
<td>Draws a full polygon (between many points) in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_FillEllipse (int Xpos, int Ypos, int XRADIUS, int YRadius)</code></td>
<td>Draws a full ellipse in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_DisplayOn (void)</code></td>
<td>Switch back on the display if was switched off by previous call of BSP_LCD_DisplayOff().</td>
</tr>
<tr>
<td><code>void BSP_LCD_DisplayOff (void)</code></td>
<td>Switch off the display.</td>
</tr>
<tr>
<td><code>void BSP_LCD_SetBrightness (uint8_t BrightnessValue)</code></td>
<td>Set the brightness value.</td>
</tr>
<tr>
<td><code>void DSI_IO_WriteCmd (uint32_t NbrParams, uint8_t</code></td>
<td>Writes a command to the display.</td>
</tr>
</tbody>
</table>
DCS or Generic short/long write command.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__weak void BSP_LCD_MspDeInit (void)</code></td>
<td>De-Initializes the BSP LCD Msp Application can surcharge if needed this function implementation.</td>
</tr>
<tr>
<td><code>__weak void BSP_LCD_MspInit (void)</code></td>
<td>Initialize the BSP LCD Msp.</td>
</tr>
<tr>
<td><code>void BSP_LCD_DrawPixel (uint16_t Xpos, uint16_t Ypos, uint32_t RGB_Code)</code></td>
<td>Draws a pixel on LCD.</td>
</tr>
</tbody>
</table>
## Variables

<table>
<thead>
<tr>
<th>Type Class</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>static DSI_VidCfgTypeDef</td>
<td>hdsivideo_handle</td>
</tr>
<tr>
<td>DMA2D_HandleTypeDef</td>
<td>hdma2d_discovery</td>
</tr>
<tr>
<td>LTDC_HandleTypeDef</td>
<td>hltdc_discovery</td>
</tr>
<tr>
<td>DSI_HandleTypeDef</td>
<td>hdsi_discovery</td>
</tr>
<tr>
<td>uint32_t</td>
<td>( \text{lcd}_x_\text{size} = \text{OTM8009A}_800X480_\text{WIDTH} )</td>
</tr>
<tr>
<td>uint32_t</td>
<td>( \text{lcd}_y_\text{size} = \text{OTM8009A}_800X480_\text{HEIGHT} )</td>
</tr>
<tr>
<td>static uint32_t</td>
<td>ActiveLayer = LTDC_ACTIVE_LAYER_BACKGROUND</td>
</tr>
</tbody>
</table>

Default Active LTDC Layer in which drawing is made is LTDC Layer Background.

<table>
<thead>
<tr>
<th>Type Class</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>static LCD_DrawPropTypeDef</td>
<td>DrawProp [LTDC_MAX_LAYER_NUMBER]</td>
</tr>
</tbody>
</table>

Current Drawing Layer properties variable.
Detailed Description

This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM32F769I-DISCOVERY board.

Author:
   MCD Application Team

Version:
   V2.0.0

Date:
   30-December-2016

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

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

Definition in file stm32f769i_discovery_lcd.c.
stm32f769i_discovery_lcd.h File Reference

This file contains the common defines and functions prototypes for the stm32469i_discovery_lcd.c driver. More...

#include "../Components/otm8009a/otm8009a.h"
#include "../Components/adv7533/adv7533.h"
#include "stm32f769i_discovery_sdram.h"
#include "stm32f769i_discovery.h"
#include "../../../Utilities/Fonts/fonts.h"
#include <string.h>

Go to the source code of this file.
# Data Structures

<table>
<thead>
<tr>
<th>struct</th>
<th><strong>LCD_DrawPropTypeDef</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LCD Drawing main properties. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>struct</th>
<th><strong>Point</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LCD Drawing point (pixel) geometric definition. <a href="#">More...</a></td>
</tr>
</tbody>
</table>
# Defines

<table>
<thead>
<tr>
<th>Define</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSP_LCD_DMA2D_IRQHandler</td>
<td>DMA2D_IRQHandler</td>
</tr>
<tr>
<td>BSP_LCD_DSI_IRQHandler</td>
<td>DSI_IRQHandler</td>
</tr>
<tr>
<td>BSP_LCD_LTDC_IRQHandler</td>
<td>LTDC_IRQHandler</td>
</tr>
<tr>
<td>BSP_LCD_LTDC_ER_IRQHandler</td>
<td>LTDC_ER_IRQHandler</td>
</tr>
<tr>
<td>LCD_LayerCfgTypeDef</td>
<td>LTDC_LayerCfgTypeDef</td>
</tr>
<tr>
<td>LCD_FB_START_ADDRESS</td>
<td>((uint32_t)0xC0000000)</td>
</tr>
<tr>
<td>LTDC_MAX_LAYER_NUMBER</td>
<td>((uint32_t)2)</td>
</tr>
<tr>
<td>LTDC_ACTIVE_LAYER_BACKGROUND</td>
<td>((uint32_t)0)</td>
</tr>
<tr>
<td>LTDC_ACTIVE_LAYER_FOREGROUND</td>
<td>((uint32_t)1)</td>
</tr>
<tr>
<td>LTDC_NB_OF_LAYERS</td>
<td>((uint32_t)2)</td>
</tr>
<tr>
<td>LTDC_DEFAULT_ACTIVE_LAYER</td>
<td>LTDC_ACTIVE_LAYER_FOREGROUND</td>
</tr>
<tr>
<td>LCD_OK</td>
<td>0x00</td>
</tr>
<tr>
<td>LCD_ERROR</td>
<td>0x01</td>
</tr>
<tr>
<td>LCD_TIMEOUT</td>
<td>0x02</td>
</tr>
<tr>
<td>LCD_OTM8009A_ID</td>
<td>((uint32_t)0)</td>
</tr>
<tr>
<td>HDMI_ADV7533_ID</td>
<td>((uint32_t)0)</td>
</tr>
<tr>
<td>HDMI_FORMAT_720_480</td>
<td>((uint8_t)0x00) /<em>720_480 format of HDMI display</em>/</td>
</tr>
<tr>
<td>HDMI_FORMAT_720_576</td>
<td>((uint8_t)0x01) /<em>720_576 format of HDMI display</em>/</td>
</tr>
<tr>
<td>LCD_COLOR_BLUE</td>
<td>((uint32_t)0xFF0000FF)</td>
</tr>
</tbody>
</table>
LCD color definitions values in ARGB8888 format.

```c
#define LCD_COLOR_GREEN  ((uint32_t) 0xFF00FF00)  
  Green value in ARGB8888 format.
#define LCD_COLOR_RED     ((uint32_t) 0xFFFF0000)  
  Red value in ARGB8888 format.
#define LCD_COLOR_CYAN    ((uint32_t) 0xFF00FFFF)  
  Cyan value in ARGB8888 format.
#define LCD_COLOR_MAGENTA ((uint32_t) 0xFFFF00FF)  
  Magenta value in ARGB8888 format.
#define LCD_COLOR_YELLOW  ((uint32_t) 0xFFFFFF00)  
  Yellow value in ARGB8888 format.
#define LCD_COLOR_LIGHTBLUE((uint32_t) 0xFF8080FF) 
  Light Blue value in ARGB8888 format.
#define LCD_COLOR_LIGHTGREEN((uint32_t) 0xFF80FF80) 
  Light Green value in ARGB8888 format.
#define LCD_COLOR_LIGHTRED ((uint32_t) 0xFFFF8080)  
  Light Red value in ARGB8888 format.
#define LCD_COLOR_LIGHTCYAN((uint32_t) 0xFF80FFFF) 
  Light Cyan value in ARGB8888 format.
#define LCD_COLOR_LIGHTMAGENTA((uint32_t) 0xFFFF80FF) 
  Light Magenta value in ARGB8888 format.
#define LCD_COLOR_LIGHTYELLOW((uint32_t) 0xFFFFFF80) 
  Light Yellow value in ARGB8888 format.
#define LCD_COLOR_DARKBLUE ((uint32_t) 0xFF000080)  
  Dark Blue value in ARGB8888 format.
#define LCD_COLOR_DARKGREEN((uint32_t) 0xFF008000)  
  Light Dark Green value in ARGB8888 format.
#define LCD_COLOR_DARKRED  ((uint32_t) 0xFF800000)  
  Light Dark Red value in ARGB8888 format.
#define LCD_COLOR_DARKCYAN ((uint32_t) 0xFF008080)  
  Dark Cyan value in ARGB8888 format.
#define LCD_COLOR_DARKMAGENTA((uint32_t) 0xFF800080) 
  Dark Magenta value in ARGB8888 format.
```
<table>
<thead>
<tr>
<th>#define</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LCD_COLOR_DARKYELLOW</strong></td>
<td>(uint32_t) 0xFF808000</td>
</tr>
<tr>
<td>Dark Yellow value in ARGB8888 format.</td>
<td></td>
</tr>
<tr>
<td><strong>LCD_COLOR_WHITE</strong></td>
<td>(uint32_t) 0xFFFFFFFF</td>
</tr>
<tr>
<td>White value in ARGB8888 format.</td>
<td></td>
</tr>
<tr>
<td><strong>LCD_COLOR_LIGHTGRAY</strong></td>
<td>(uint32_t) 0xFFD3D3D3</td>
</tr>
<tr>
<td>Light Gray value in ARGB8888 format.</td>
<td></td>
</tr>
<tr>
<td><strong>LCD_COLOR_GRAY</strong></td>
<td>(uint32_t) 0xFF808080</td>
</tr>
<tr>
<td>Gray value in ARGB8888 format.</td>
<td></td>
</tr>
<tr>
<td><strong>LCD_COLOR_DARKGRAY</strong></td>
<td>(uint32_t) 0xFF404040</td>
</tr>
<tr>
<td>Dark Gray value in ARGB8888 format.</td>
<td></td>
</tr>
<tr>
<td><strong>LCD_COLOR_BLACK</strong></td>
<td>(uint32_t) 0xFF000000</td>
</tr>
<tr>
<td>Black value in ARGB8888 format.</td>
<td></td>
</tr>
<tr>
<td><strong>LCD_COLOR_BROWN</strong></td>
<td>(uint32_t) 0xFFA52A2A</td>
</tr>
<tr>
<td>Brown value in ARGB8888 format.</td>
<td></td>
</tr>
<tr>
<td><strong>LCD_COLOR_ORANGE</strong></td>
<td>(uint32_t) 0xFFFFA500</td>
</tr>
<tr>
<td>Orange value in ARGB8888 format.</td>
<td></td>
</tr>
<tr>
<td><strong>LCD_COLOR_TRANSPARENT</strong></td>
<td>(uint32_t) 0xFF000000</td>
</tr>
<tr>
<td>Transparent value in ARGB8888 format.</td>
<td></td>
</tr>
<tr>
<td><strong>LCD_DEFAULT_FONT</strong></td>
<td>Font24</td>
</tr>
<tr>
<td>LCD default font.</td>
<td></td>
</tr>
<tr>
<td><strong>LCD_DSI_PIXEL_DATA_FMT_RBG888</strong></td>
<td>DSI_RGB888</td>
</tr>
<tr>
<td>Possible values of pixel data format (ie color coding) transmitted on DSI Data lane in DSI packets.</td>
<td></td>
</tr>
<tr>
<td><strong>LCD_DSI_PIXEL_DATA_FMT_RBG565</strong></td>
<td>DSI_RGB565</td>
</tr>
</tbody>
</table>
Typedefs

typedef `Point` * `pPoint`  
Pointer on LCD Drawing point (pixel) geometric definition.
## Enumerations

<table>
<thead>
<tr>
<th>Enum</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Text_AlignModeTypeDef</strong></td>
<td><code>CENTER_MODE = 0x01, RIGHT_MODE = 0x02, LEFT_MODE = 0x03</code></td>
</tr>
<tr>
<td>LCD drawing Line alignment mode definitions.</td>
<td></td>
</tr>
<tr>
<td><strong>LCD_OrientationTypeDef</strong></td>
<td><code>LCD_ORIENTATION_PORTRAIT = 0x00, LCD_ORIENTATION_LANDSCAPE = 0x01, LCD_ORIENTATION_INVALID = 0x02</code></td>
</tr>
<tr>
<td>LCD_OrientationTypeDef Possible values of Display Orientation.</td>
<td></td>
</tr>
</tbody>
</table>
## Functions

<table>
<thead>
<tr>
<th>uint8_t</th>
<th><strong>BSP_LCD_Init</strong> (void)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initializes the DSI LCD.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>uint8_t</th>
<th><strong>BSP_LCD_InitEx</strong> (LCD_OrientationTypeDef orientation)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initializes the DSI LCD.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>__weak void</th>
<th><strong>BSP_LCD_MspDeInit</strong> (void)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>De-Initializes the BSP LCD Msp Application can surcharge if needed this function implementation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>__weak void</th>
<th><strong>BSP_LCD_MspInit</strong> (void)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initialize the BSP LCD Msp.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>BSP_LCD_Reset</strong> (void)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BSP LCD Reset Hw reset the LCD DSI activating its XRES signal (active low for some time) and desactivating it later.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>uint32_t</th>
<th><strong>BSP_LCD_GetXSize</strong> (void)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the LCD X size.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>uint32_t</th>
<th><strong>BSP_LCD_GetYSize</strong> (void)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the LCD Y size.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>BSP_LCD_SetXSize</strong> (uint32_t imageWidthPixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set the LCD X size.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>BSP_LCD_SetYSize</strong> (uint32_t imageHeightPixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set the LCD Y size.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>BSP_LCD_LayerDefaultInit</strong> (uint16_t LayerIndex, uint32_t FB_Address)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initializes the LCD layers.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>BSP_LCD_SetTransparency</strong> (uint32_t LayerIndex, uint8_t Transparency)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Configures the transparency.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>BSP_LCD_SetLayerAddress</strong> (uint32_t LayerIndex, uint32_t Address)</th>
</tr>
</thead>
</table>
Sets an LCD layer frame buffer address.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void BSP_LCD_SetColorKeying</code></td>
<td>Configures and sets the color keying.</td>
</tr>
<tr>
<td><code>void BSP_LCD_ResetColorKeying</code></td>
<td>Disables the color keying.</td>
</tr>
<tr>
<td><code>void BSP_LCD_SetLayerWindow</code></td>
<td>Sets display window.</td>
</tr>
<tr>
<td><code>void BSP_LCD_SelectLayer</code></td>
<td>Selects the LCD Layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_SetLayerVisible</code></td>
<td>Sets an LCD Layer visible.</td>
</tr>
<tr>
<td><code>void BSP_LCD_SetTextColor</code></td>
<td>Sets the LCD text color.</td>
</tr>
<tr>
<td><code>uint32_t BSP_LCD_GetTextColor</code></td>
<td>Gets the LCD text color.</td>
</tr>
<tr>
<td><code>void BSP_LCD_SetBackColor</code></td>
<td>Sets the LCD background color.</td>
</tr>
<tr>
<td><code>uint32_t BSP_LCD_GetBackColor</code></td>
<td>Gets the LCD background color.</td>
</tr>
<tr>
<td><code>void BSP_LCD_GetFont</code></td>
<td>Gets the LCD text font.</td>
</tr>
<tr>
<td><code>sFONT * BSP_LCD_GetFont</code></td>
<td>Gets the LCD text font.</td>
</tr>
<tr>
<td><code>uint32_t BSP_LCD_ReadPixel</code></td>
<td>Reads an LCD pixel.</td>
</tr>
<tr>
<td><code>void BSP_LCD_DrawPixel</code></td>
<td>Draws a pixel on LCD.</td>
</tr>
<tr>
<td><code>void BSP_LCD_Clear</code></td>
<td>Clears the whole currently active layer of LTDC.</td>
</tr>
</tbody>
</table>

Sets display window.
<table>
<thead>
<tr>
<th>void</th>
<th><strong>BSP_LCD_ClearStringLine</strong> (uint32_t Line)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clears the selected line in currently active layer.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>BSP_LCD_DisplayStringAtLine</strong> (uint16_t Line, uint8_t *ptr)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Displays a maximum of 60 characters on the LCD.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>BSP_LCD_DisplayStringAt</strong> (uint16_t Xpos, uint16_t Ypos, uint8_t *Text, Text_AlignModeTypdef Mode)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Displays characters in currently active layer.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>BSP_LCD_DisplayChar</strong> (uint16_t Xpos, uint16_t Ypos, uint8_t Ascii)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Displays one character in currently active layer.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>BSP_LCD_DrawHLine</strong> (uint16_t Xpos, uint16_t Ypos, uint16_t Length)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Draws a horizontal line in currently active layer.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>BSP_LCD_DrawVLine</strong> (uint16_t Xpos, uint16_t Ypos, uint16_t Length)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Draws a vertical line in currently active layer.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>BSP_LCD_DrawLine</strong> (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Draws a uni-line (between two points) in currently active layer.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>BSP_LCD_DrawRect</strong> (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Draws a rectangle in currently active layer.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>BSP_LCD_DrawCircle</strong> (uint16_t Xpos, uint16_t Ypos, uint16_t Radius)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Draws a circle in currently active layer.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>BSP_LCD_DrawPolygon</strong> (pPoint Points, uint16_t PointCount)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Draws an poly-line (between many points) in currently active layer.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>BSP_LCD_DrawEllipse</strong> (int Xpos, int Ypos, int XRadius, int YRadius)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Draws an ellipse on LCD in currently active layer.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void BSP_LCD_DrawBitmap (uint32_t Xpos, uint32_t Ypos, uint8_t *pbmp)</code></td>
<td>Draws a bitmap picture loaded in the internal Flash (32 bpp) in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_FillRect (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height)</code></td>
<td>Draws a full rectangle in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_FillCircle (uint16_t Xpos, uint16_t Ypos, uint16_t Radius)</code></td>
<td>Draws a full circle in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_FillPolygon (pPoint Points, uint16_t PointCount)</code></td>
<td>Draws a full poly-line (between many points) in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_FillEllipse (int Xpos, int Ypos, int XRadius, int YRadius)</code></td>
<td>Draws a full ellipse in currently active layer.</td>
</tr>
<tr>
<td><code>void BSP_LCD_DisplayOn (void)</code></td>
<td>Switch back on the display if was switched off by previous call of <code>BSP_LCD_DisplayOff()</code>.</td>
</tr>
<tr>
<td><code>void BSP_LCD_DisplayOff (void)</code></td>
<td>Switch Off the display.</td>
</tr>
<tr>
<td><code>void BSP_LCD_SetBrightness (uint8_t BrightnessValue)</code></td>
<td>Set the brightness value.</td>
</tr>
</tbody>
</table>
Variables

| DMA2D_HandleTypeDef | hdma2d_discovery |
**Detailed Description**

This file contains the common defines and functions prototypes for the
stm32469i_discovery_lcd.c driver.

**Author:**
MCD Application Team

**Version:**
V2.0.0

**Date:**
30-December-2016

**Attention:**
© COPYRIGHT(c) 2016 STMicroelectronics

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

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

Definition in file stm32f769i_discovery_lcd.h.

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
STM32F769I-Discovery BSP User Manual

stm32f769i_discovery_qspi.c File Reference

This file includes a standard driver for the MX25L512 QSPI memory mounted on STM32F769I-Discovery board. More...

#include "stm32f769i_discovery_qspi.h"

Go to the source code of this file.
## Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>static uint8_t QSPI_ResetMemory</strong></td>
<td>(QSPI_HandleTypeDef *hqspi) This function reset the QSPI memory.</td>
</tr>
<tr>
<td><strong>static uint8_t QSPI_EnterFourBytesAddress</strong></td>
<td>(QSPI_HandleTypeDef *hqspi) This function set the QSPI memory in 4-byte address mode.</td>
</tr>
<tr>
<td><strong>static uint8_t QSPI_DummyCyclesCfg</strong></td>
<td>(QSPI_HandleTypeDef *hqspi) This function configure the dummy cycles on memory side.</td>
</tr>
<tr>
<td><strong>static uint8_t QSPI_EnterMemory_QPI</strong></td>
<td>(QSPI_HandleTypeDef *hqspi) This function put QSPI memory in QPI mode (quad I/O).</td>
</tr>
<tr>
<td><strong>static uint8_t QSPI_ExitMemory_QPI</strong></td>
<td>(QSPI_HandleTypeDef *hqspi) This function put QSPI memory in SPI mode.</td>
</tr>
<tr>
<td><strong>static uint8_t QSPI_OutDrvStrengthCfg</strong></td>
<td>(QSPI_HandleTypeDef *hqspi) This function configure the Output driver strength on memory side.</td>
</tr>
<tr>
<td><strong>static uint8_t QSPI_WriteEnable</strong></td>
<td>(QSPI_HandleTypeDef *hqspi) This function send a Write Enable and wait it is effective.</td>
</tr>
<tr>
<td><strong>static uint8_t QSPI_AutoPollingMemReady</strong></td>
<td>(QSPI_HandleTypeDef *hqspi, uint32_t Timeout) This function read the SR of the memory and wait the EOP.</td>
</tr>
<tr>
<td><strong>uint8_t BSP_QSPI_Init</strong></td>
<td>(void) Initializes the QSPI interface.</td>
</tr>
<tr>
<td><strong>uint8_t BSP_QSPI_DeInit</strong></td>
<td>(void) De-Initializes the QSPI interface.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td><code>BSP_QSPI_Read</code></td>
<td><code>uint8_t</code> <code>BSP_QSPI_Read</code> (uint8_t *pData, uint32_t ReadAddr, uint32_t Size) Reads an amount of data from the QSPI memory.</td>
</tr>
<tr>
<td><code>BSP_QSPI_Write</code></td>
<td><code>uint8_t</code> <code>BSP_QSPI_Write</code> (uint8_t *pData, uint32_t WriteAddr, uint32_t Size) Writes an amount of data to the QSPI memory.</td>
</tr>
<tr>
<td><code>BSP_QSPI_Erase_Block</code></td>
<td><code>uint8_t</code> <code>BSP_QSPI_Erase_Block</code> (uint32_t BlockAddress) Erases the specified block of the QSPI memory.</td>
</tr>
<tr>
<td><code>BSP_QSPI_Erase_Chip</code></td>
<td><code>uint8_t</code> <code>BSP_QSPI_Erase_Chip</code> (void) Erases the entire QSPI memory.</td>
</tr>
<tr>
<td><code>BSP_QSPI_GetStatus</code></td>
<td><code>uint8_t</code> <code>BSP_QSPI_GetStatus</code> (void) Reads current status of the QSPI memory.</td>
</tr>
<tr>
<td><code>BSP_QSPI_GetInfo</code></td>
<td><code>uint8_t</code> <code>BSP_QSPI_GetInfo</code> (QSPI_Info *pInfo) Return the configuration of the QSPI memory.</td>
</tr>
<tr>
<td><code>BSP_QSPI_EnableMemoryMappedMode</code></td>
<td><code>uint8_t</code> <code>BSP_QSPI_EnableMemoryMappedMode</code> (void) Configure the QSPI in memory-mapped mode.</td>
</tr>
<tr>
<td><code>__weak void BSP_QSPI_MspInit</code></td>
<td><code>__weak void</code> <code>BSP_QSPI_MspInit</code> (QSPI_HandleTypeDef *hqspi, void *Params) QSPI MSP Initialization This function configures the hardware resources used in this example:</td>
</tr>
<tr>
<td><code>__weak void BSP_QSPI_MspDeInit</code></td>
<td><code>__weak void</code> <code>BSP_QSPI_MspDeInit</code> (QSPI_HandleTypeDef *hqspi, void *Params) QSPI MSP De-Initialization This function frees the hardware resources used in this example:</td>
</tr>
</tbody>
</table>
Variables

| QSPI_HandleTypeDef | QSPIHandle |
Detailed Description

This file includes a standard driver for the MX25L512 QSPI memory mounted on STM32F769I-Discovery board.

Author:
MCD Application Team

Version:
V2.0.0

Date:
30-December-2016

[..]

(#) This driver is used to drive the MX25L512 QSPI external memory mounted on STM32F769I-Discovery board.

(#) This driver need a specific component driver (MX25L51245G) to be included with.

(#) Initialization steps:
   (++) Initialize the QSPI external memory using the BSP_QSPI_Init() function. This function includes the MSP layer hardware resources initialization and the QSPI interface with the external memory.
QSPI memory operations

QSPI memory can be accessed with read/write operations once it is initialized.

Read/write operation can be performed with AHB access using the functions
BSP_QSPI_Read()/BSP_QSPI_Write().

The function BSP_QSPI_GetInfo() returns the configuration of the QSPI memory.
(see the QSPI memory data sheet)

Perform erase block operation using the function BSP_QSPI_Erase_Block() and by specifying the block address. You can perform an erase operation of the whole chip by calling the function BSP_QSPI_Erase_Chip().

The function BSP_QSPI_GetStatus() returns the current status of the QSPI memory.
(see the QSPI memory data sheet)

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

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

Definition in file stm32f769i_discovery_qspi.c.
This file contains the common defines and functions prototypes for the `stm32f769i_discovery_qspi.c` driver. More...

```
#include "stm32f7xx_hal.h" #include ".../Components/mx25l512/mx25l512.h"
```

Go to the source code of this file.
Data Structures

struct QSPI_Info
Defines

#define QSPI_OK ((uint8_t)0x00)
#define QSPI_ERROR ((uint8_t)0x01)
#define QSPI_BUSY ((uint8_t)0x02)
#define QSPI_NOT_SUPPORTED ((uint8_t)0x04)
#define QSPI_SUSPENDED ((uint8_t)0x08)
#define QSPI_CLK_ENABLE() __HAL_RCC_QSPI_CLK_ENABLE()
#define QSPI_CLK_DISABLE() __HAL_RCC_QSPI_CLK_DISABLE()
#define QSPI_CS_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_ENABLE()
#define QSPI_CLK_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_ENABLE()
#define QSPI_D0_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define QSPI_D1_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define QSPI_D2_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define QSPI_D3_GPIO_CLK_ENABLE() __HAL_RCC_GPIOD_CLK_ENABLE()
#define QSPI_FORCE_RESET() __HAL_RCC_QSPI_FORCE_RESET()
#define QSPI_RELEASE_RESET() __HAL_RCC_QSPI_RELEASE_RESET()
#define QSPI_CS_PIN GPIO_PIN_6
#define QSPI_CS_GPIO_PORT GPIOB
#define QSPI_CS_PIN_AF GPIO_AF10_QUADSPI
#define QSPI_CLK_PIN GPIO_PIN_2
#define QSPI_CLK_GPIO_PORT GPIOB
#define QSPI_CLK_PIN_AF GPIO_AF9_QUADSPI
#define QSPI_D0_PIN GPIO_PIN_9
#define QSPI_D0_GPIO_PORT GPIOC
#define QSPI_D0_PIN_AF GPIO_AF9_QUADSPI
#define QSPI_D1_PIN GPIO_PIN_10
#define QSPI_D1_GPIO_PORT GPIOC
#define QSPI_D1_PIN_AF GPIO_AF9_QUADSPI
#define QSPI_D2_PIN GPIO_PIN_2
#define QSPI_D2_GPIO_PORT GPIOE
#define QSPI_D2_PIN_AF GPIO_AF9_QUADSPI
#define QSPI_D3_PIN GPIO_PIN_13
#define QSPI_D3_GPIO_PORT GPIOD
#define QSPI_D3_PIN_AF GPIO_AF9_QUADSPI
## Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BSP_QSPI_Init</code></td>
<td>Initializes the QSPI interface.</td>
</tr>
<tr>
<td><code>BSP_QSPI_DeInit</code></td>
<td>De-Initializes the QSPI interface.</td>
</tr>
<tr>
<td><code>BSP_QSPI_Read</code></td>
<td>Reads an amount of data from the QSPI memory.</td>
</tr>
<tr>
<td><code>BSP_QSPI_Write</code></td>
<td>Writes an amount of data to the QSPI memory.</td>
</tr>
<tr>
<td><code>BSP_QSPI_Erase_Block</code></td>
<td>Erases the specified block of the QSPI memory.</td>
</tr>
<tr>
<td><code>BSP_QSPI_Erase_Chip</code></td>
<td>Erases the entire QSPI memory.</td>
</tr>
<tr>
<td><code>BSP_QSPI_GetStatus</code></td>
<td>Reads current status of the QSPI memory.</td>
</tr>
<tr>
<td><code>BSP_QSPI_GetInfo</code></td>
<td>Returns the configuration of the QSPI memory.</td>
</tr>
<tr>
<td><code>BSP_QSPI_EnableMemoryMappedMode</code></td>
<td>Configure the QSPI in memory-mapped mode.</td>
</tr>
<tr>
<td><code>BSP_QSPI_MspInit</code></td>
<td>QSPI MSP Initialization This function configures the hardware resources used in this example:</td>
</tr>
<tr>
<td><code>BSP_QSPI_MspDeInit</code></td>
<td>QSPI MSP De-Initialization This function frees the hardware resources used in this example:</td>
</tr>
</tbody>
</table>
Detailed Description

This file contains the common defines and functions prototypes for the
stm32f769i_discovery_qspi.c driver.

Author:
   MCD Application Team

Version:
   V2.0.0

Date:
   30-December-2016

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

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

Definition in file `stm32f769i_discovery_qspi.h`.

---

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery
BSP User Manual by doxygen 1.7.6.1
STM32F769I-Discovery BSP User Manual

stm32f769i_discovery_sd.c File Reference

This file includes the uSD card driver mounted on STM32F769I-Discovery board. More...

#include "stm32f769i_discovery_sd.h"

Go to the source code of this file.
## Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uint8_t BSP_SD_Init (void)</code></td>
<td>Initializes the SD card device.</td>
</tr>
<tr>
<td><code>uint8_t BSP_SD_DeInit (void)</code></td>
<td>DeInitializes the SD card device.</td>
</tr>
<tr>
<td><code>uint8_t BSP_SD_ITConfig (void)</code></td>
<td>Configures Interrupt mode for SD detection pin.</td>
</tr>
<tr>
<td><code>uint8_t BSP_SD_IsDetected (void)</code></td>
<td>Detects if SD card is correctly plugged in the memory slot or not.</td>
</tr>
<tr>
<td><code>uint8_t BSP_SD_ReadBlocks (uint32_t *pData, uint32_t ReadAddr, uint32_t NumOfBlocks, uint32_t Timeout)</code></td>
<td>Reads block(s) from a specified address in an SD card, in polling mode.</td>
</tr>
<tr>
<td><code>uint8_t BSP_SD_WriteBlocks (uint32_t *pData, uint32_t WriteAddr, uint32_t NumOfBlocks, uint32_t Timeout)</code></td>
<td>Writes block(s) to a specified address in an SD card, in polling mode.</td>
</tr>
<tr>
<td><code>uint8_t BSP_SD_ReadBlocks_DMA (uint32_t *pData, uint32_t ReadAddr, uint32_t NumOfBlocks)</code></td>
<td>Reads block(s) from a specified address in an SD card, in DMA mode.</td>
</tr>
<tr>
<td><code>uint8_t BSP_SD_WriteBlocks_DMA (uint32_t *pData, uint32_t WriteAddr, uint32_t NumOfBlocks)</code></td>
<td>Writes block(s) to a specified address in an SD card, in DMA mode.</td>
</tr>
<tr>
<td><code>uint8_t BSP_SD_Erase (uint32_t StartAddr, uint32_t EndAddr)</code></td>
<td>Erases the specified memory area of the given SD card.</td>
</tr>
<tr>
<td><code>__weak void BSP_SD_MspInit (SD_HandleTypeDef *hsd, void *Params)</code></td>
<td>Initializes the SD MSP.</td>
</tr>
<tr>
<td>Function Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>__weak void BSP_SD_Detect_MspInit (SD_HandleTypeDef *hsd, void *Params)</td>
<td>Initializes the SD Detect pin MSP.</td>
</tr>
<tr>
<td>__weak void BSP_SD_MspDeInit (SD_HandleTypeDef *hsd, void *Params)</td>
<td>DeInitializes the SD MSP.</td>
</tr>
<tr>
<td>uint8_t BSP_SD_GetCardState (void)</td>
<td>Gets the current SD card data status.</td>
</tr>
<tr>
<td>void BSP_SD_GetCardInfo (HAL_SD_CardInfoTypeDef *CardInfo)</td>
<td>Get SD information about specific SD card.</td>
</tr>
<tr>
<td>void HAL_SD_AbortCallback (SD_HandleTypeDef *hsd)</td>
<td>SD Abort callbacks.</td>
</tr>
<tr>
<td>void HAL_SD_TxCpltCallback (SD_HandleTypeDef *hsd)</td>
<td>Tx Transfer completed callbacks.</td>
</tr>
<tr>
<td>void HAL_SD_RxCpltCallback (SD_HandleTypeDef *hsd)</td>
<td>Rx Transfer completed callbacks.</td>
</tr>
<tr>
<td>__weak void BSP_SD_AbortCallback (void)</td>
<td>BSP SD Abort callbacks.</td>
</tr>
<tr>
<td>__weak void BSP_SD_WriteCpltCallback (void)</td>
<td>BSP Tx Transfer completed callbacks.</td>
</tr>
<tr>
<td>__weak void BSP_SD_ReadCpltCallback (void)</td>
<td>BSP Rx Transfer completed callbacks.</td>
</tr>
</tbody>
</table>
Variables

<table>
<thead>
<tr>
<th>SD_HandleTypeDef</th>
<th>uSdHandle</th>
</tr>
</thead>
</table>

Detailed Description

This file includes the uSD card driver mounted on STM32F769I-Discovery board.

Author:
MCD Application Team

Version:
V2.0.0

Date:
30-December-2016

Attention:
© COPYRIGHT(c) 2016 STMicroelectronics

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

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

Definition in file stdf769i_discovery_sd.c.
stm32f769i_discovery_sd.h File Reference

This file contains the common defines and functions prototypes for the stm32f769i_discovery_sd.c driver. More...

#include "stm32f769i_discovery.h"

Go to the source code of this file.
### Defines

<table>
<thead>
<tr>
<th>Define</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSP_SD_CardInfo</td>
<td>HAL_SD_CardInfoTypeDef</td>
</tr>
<tr>
<td>MSD_OK</td>
<td>((uint8_t)0x00)</td>
</tr>
<tr>
<td>MSD_ERROR</td>
<td>((uint8_t)0x01)</td>
</tr>
<tr>
<td>MSD_ERROR_SD_NOT_PRESENT</td>
<td>((uint8_t)0x02)</td>
</tr>
<tr>
<td>SD_TRANSFER_OK</td>
<td>((uint8_t)0x00)</td>
</tr>
<tr>
<td>SD_TRANSFER_BUSY</td>
<td>((uint8_t)0x01)</td>
</tr>
<tr>
<td>SD_PRESENT</td>
<td>((uint8_t)0x01)</td>
</tr>
<tr>
<td>SD_NOT_PRESENT</td>
<td>((uint8_t)0x00)</td>
</tr>
<tr>
<td>SD_DATATIMEOUT</td>
<td>((uint32_t)100000000)</td>
</tr>
<tr>
<td>__DMAx_TxRx_CLK_ENABLE</td>
<td>__HAL_RCC_DMA2_CLK_ENABLE</td>
</tr>
<tr>
<td>SD_DMAx_Tx_CHANNEL</td>
<td>DMA_CHANNEL_11</td>
</tr>
<tr>
<td>SD_DMAx_Rx_CHANNEL</td>
<td>DMA_CHANNEL_11</td>
</tr>
<tr>
<td>SD_DMAx_Tx_STREAM</td>
<td>DMA2_Stream5</td>
</tr>
<tr>
<td>SD_DMAx_Rx_STREAM</td>
<td>DMA2_Stream0</td>
</tr>
<tr>
<td>SD_DMAx_Tx_IRQn</td>
<td>DMA2_Stream5_IRQHandler</td>
</tr>
<tr>
<td>SD_DMAx_Rx_IRQn</td>
<td>DMA2_Stream0_IRQHandler</td>
</tr>
<tr>
<td>BSP_SDMMC_IRQHandler</td>
<td>SDMMC2_IRQHandler</td>
</tr>
<tr>
<td>BSP_SDMMC_DMA_Tx_IRQHandler</td>
<td>DMA2_Stream5_IRQHandler</td>
</tr>
<tr>
<td>BSP_SDMMC_DMA_Rx_IRQHandler</td>
<td>DMA2_Stream0_IRQHandler</td>
</tr>
<tr>
<td>SD_DetectIRQHandler</td>
<td>HAL_GPIO_EXTI_IRQHandler(SD_DETECT_PIN)</td>
</tr>
</tbody>
</table>
## Functions

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_Init</strong> (void)</td>
<td>Initializes the SD card device.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_DeInit</strong> (void)</td>
<td>Deinitializes the SD card device.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_ITConfig</strong> (void)</td>
<td>Configures Interrupt mode for SD detection pin.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_ReadBlocks</strong> (uint32_t *pData, uint32_t ReadAddr, uint32_t NumOfBlocks, uint32_t Timeout)</td>
<td>Reads block(s) from a specified address in an SD card, in polling mode.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_WriteBlocks</strong> (uint32_t *pData, uint32_t WriteAddr, uint32_t NumOfBlocks, uint32_t Timeout)</td>
<td>Writes block(s) to a specified address in an SD card, in polling mode.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_ReadBlocks_DMA</strong> (uint32_t *pData, uint32_t ReadAddr, uint32_t NumOfBlocks)</td>
<td>Reads block(s) from a specified address in an SD card, in DMA mode.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_WriteBlocks_DMA</strong> (uint32_t *pData, uint32_t WriteAddr, uint32_t NumOfBlocks)</td>
<td>Writes block(s) to a specified address in an SD card, in DMA mode.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_Erase</strong> (uint32_t StartAddr, uint32_t EndAddr)</td>
<td>Erases the specified memory area of the given SD card.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_GetCardState</strong> (void)</td>
<td>Gets the current SD card data status.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><strong>BSP_SD_GetCardInfo</strong> (HAL_SD_CardInfoTypeDef *CardInfo)</td>
<td>Get SD information about specific SD card.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_IsDetected</strong> (void)</td>
<td>Detects if SD card is correctly plugged in the memory slot or</td>
</tr>
<tr>
<td>Function Name</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>void BSP_SD_MspInit (SD_HandleTypeDef *hsd, void *Params)</code></td>
<td>Initializes the SD MSP.</td>
<td></td>
</tr>
<tr>
<td><code>void BSP_SD_Detect_MspInit (SD_HandleTypeDef *hsd, void *Params)</code></td>
<td>Initializes the SD Detect pin MSP.</td>
<td></td>
</tr>
<tr>
<td><code>void BSP_SD_MspDeInit (SD_HandleTypeDef *hsd, void *Params)</code></td>
<td>DeInitializes the SD MSP.</td>
<td></td>
</tr>
<tr>
<td><code>void BSP_SD_AbortCallback (void)</code></td>
<td>BSP SD Abort callbacks.</td>
<td></td>
</tr>
<tr>
<td><code>void BSP_SD_WriteCpltCallback (void)</code></td>
<td>BSP Tx Transfer completed callbacks.</td>
<td></td>
</tr>
<tr>
<td><code>void BSP_SD_ReadCpltCallback (void)</code></td>
<td>BSP Rx Transfer completed callbacks.</td>
<td></td>
</tr>
</tbody>
</table>
Detailed Description

This file contains the common defines and functions prototypes for the stm32f769i_discovery_sd.c driver.

Author:
MCD Application Team

Version:
V2.0.0

Date:
30-December-2016

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

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

Definition in file stm32f769i_discovery_sd.h.
stm32f769i_discovery_sdram.c File Reference

This file includes the SDRAM driver for the MT48LC4M32B2B5-6A memory device mounted on STM32F769I-DISCOVERY boards. More...

#include "stm32f769i_discovery_sdram.h"

Go to the source code of this file.
## Functions

<table>
<thead>
<tr>
<th>data_type</th>
<th>function_name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td>BSP_SDRAM_Init (void)</td>
<td>Initializes the SDRAM device.</td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_SDRAM_DeInit (void)</td>
<td>Deinitializes the SDRAM device.</td>
</tr>
<tr>
<td>void</td>
<td>BSP_SDRAM_Initialization_sequence (uint32_t RefreshCount)</td>
<td>Programs the SDRAM device.</td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_SDRAM_ReadData (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)</td>
<td>Reads an amount of data from the SDRAM memory in polling mode.</td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_SDRAM_ReadData_DMA (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)</td>
<td>Reads an amount of data from the SDRAM memory in DMA mode.</td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_SDRAM_WriteData (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)</td>
<td>Writes an amount of data to the SDRAM memory in polling mode.</td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_SDRAM_WriteData_DMA (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)</td>
<td>Writes an amount of data to the SDRAM memory in DMA mode.</td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_SDRAM_Sendcmd (FMC_SDRAM_CommandTypeDef *SdramCmd)</td>
<td>Sends command to the SDRAM bank.</td>
</tr>
<tr>
<td>__weak</td>
<td>BSP_SDRAM_MspInit (SDRAM_HandleTypeDef *hsdram, void *Params)</td>
<td>Initializes SDRAM MSP.</td>
</tr>
<tr>
<td>__weak</td>
<td>BSP_SDRAM_MspDeInit (SDRAM_HandleTypeDef *hsdram, void *Params)</td>
<td>Deinitializes SDRAM MSP.</td>
</tr>
</tbody>
</table>
### Variables

<table>
<thead>
<tr>
<th>SDRAM_HandleTypeDef</th>
<th>sDRAMHandle</th>
</tr>
</thead>
<tbody>
<tr>
<td>static FMC_SDRAM_TimingTypeDef</td>
<td>Timing</td>
</tr>
<tr>
<td>static FMC_SDRAM_CommandTypeDef</td>
<td>Command</td>
</tr>
</tbody>
</table>
Detailed Description

This file includes the SDRAM driver for the MT48LC4M32B2B5-6A memory device mounted on STM32F769I-DISCOVERY boards.

Author:
MCD Application Team

Version:
V2.0.0

Date:
30-December-2016

How To use this driver:
-----------------------
- This driver is used to drive the MT48LC4M32B2B5-6A SDRAM external memory mounted on STM32F769I-DISCOVERY board.
- This driver does not need a specific component driver for the SDRAM device to be included with.

Driver description:
------------------
+ Initialization steps:
  o Initialize the SDRAM external memory using the BSP_SDRAM_Init() function. This function includes the MSP layer hardware resources initialization and the FMC controller configuration to interface with the external SDRAM memory.
  o It contains the SDRAM initialization sequence to program the SDRAM external device using the function BSP_SDRAM_Initialization_sequence(). Note that this sequence is standard for all SDRAM devi
ces, but can include some differences from a device to another. If it is the case, the right sequence should be implemented separately.

+ SDRAM read/write operations
  o SDRAM external memory can be accessed with read/write operations once it is initialized.
    Read/write operation can be performed with AHB access using the functions
    BSP_SDRAM_ReadData()/BSP_SDRAM_WriteData(), or by DMA transfer using the functions
    BSP_SDRAM_ReadData_DMA()/BSP_SDRAM_WriteData_DMA().
  o The AHB access is performed with 32-bit width transaction, the DMA transfer configuration is fixed at single (no burst) word transfer (see the SDRAM_MspInit() static function).
  o User can implement his own functions for read/write access with his desired configurations.
  o If interrupt mode is used for DMA transfer, the function BSP_SDRAM_DMA_IRQHandler() is called in IRQ handler file, to serve the generated interrupt once the DMA transfer is complete.
  o You can send a command to the SDRAM device in runtime using the function
    BSP_SDRAM_Sendcmd(), and giving the desired command as parameter chosen between the predefined commands of the "FMC_SDRAM_CommandTypeDef" structure.

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

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

Definition in file stm32f769i_discovery_sDRAM.c.
stm32f769i_discovery_sdram.h File Reference

This file contains the common defines and functions prototypes for the stm32f769i_discovery_sdram.c driver. More...

#include "stm32f7xx_hal.h"

Go to the source code of this file.
Defines

```c
#define SDRAM_OK     ((uint8_t)0x00)
SDRAM status structure definition.
#define SDRAM_ERROR  ((uint8_t)0x01)
#define SDRAM_DEVICE_ADDR  ((uint32_t)0xC0000000)
#define SDRAM_DEVICE_SIZE  ((uint32_t)0x1000000) /* SDRAM device size */
#define SDRAM_MEMORY_WIDTH  FMC_SDRAM_MEM_BUS_WIDTH_32
#define SDRAM_TIMEOUT  ((uint32_t)0xFFFF)
#define __DMAx_CLK_ENABLE  __HAL_RCC_DMA2_CLK_ENABLE
#define __DMAx_CLK_DISABLE  __HAL_RCC_DMA2_CLK_DISABLE
#define SDRAM_DMAx_CHANNEL  DMA_CHANNEL_0
#define SDRAM_DMAx_Stream  DMA2_Stream0
#define SDRAM_DMAx_IRQn  DMA2_Stream0_IRQn
#define BSP_SDRAM_DMA_IRQHandler  DMA2_Stream0_IRQHandler
#define SDRAM_MODEREG_BURST_LENGTH_1  ((uint16_t)0x0000)
FMC SDRAM Mode definition register defines.
#define SDRAM_MODEREG_BURST_LENGTH_2  ((uint16_t)0x0001)
#define SDRAM_MODEREG_BURST_LENGTH_4  ((uint16_t)0x0002)
#define SDRAM_MODEREG_BURST_LENGTH_8  ((uint16_t)0x0004)
#define SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL  ((uint16_t)0x0000)
#define SDRAM_MODEREG_BURST_TYPE_INTERLEAVED  ((uint16_t)0x0008)
#define SDRAM_MODEREG_CAS_LATENCY_2  ((uint16_t)0x0020)
#define SDRAM_MODEREG_CAS_LATENCY_3  ((uint16_t)0x0030)
#define SDRAM_MODEREG_OPERATING_MODE_STANDARD  ((uint16_t)0x0000)
#define SDRAM_MODEREG_WRITEBURST_MODE_PROGRAMMED  ((uint16_t)0x0000)
#define SDRAM_MODEREG_WRITEBURST_MODE_SINGLE  ((uint16_t)0x0004)
```
## Functions

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td><strong>BSP_SDRAM_Init</strong> (void)</td>
<td>Initializes the SDRAM device.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_SDRAM_DeInit</strong> (void)</td>
<td>DeInitializes the SDRAM device.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_SDRAM_Initiation_sequence</strong> (uint32_t RefreshCount)</td>
<td>Programs the SDRAM device.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_SDRAM_ReadData</strong> (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)</td>
<td>Reads an amount of data from the SDRAM memory in polling mode.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_SDRAM_ReadData_DMA</strong> (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)</td>
<td>Reads an amount of data from the SDRAM memory in DMA mode.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_SDRAM_WriteData</strong> (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)</td>
<td>Writes an amount of data to the SDRAM memory in polling mode.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_SDRAM_WriteData_DMA</strong> (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)</td>
<td>Writes an amount of data to the SDRAM memory in DMA mode.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_SDRAM_Sendcmd</strong> (FMC_SDRAM_CommandTypeDef *SdramCmd)</td>
<td>Sends command to the SDRAM bank.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_SDRAM_MspInit</strong> (SDRAM_HandleTypeDef *hsdram, void *Params)</td>
<td>Initializes SDRAM MSP.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_SDRAM_MspDeInit</strong> (SDRAM_HandleTypeDef *hsdram, void *Params)</td>
<td>DeInitializes SDRAM MSP.</td>
</tr>
</tbody>
</table>
Detailed Description

This file contains the common defines and functions prototypes for the stm32f769i_discovery_sdram.c driver.

Author:
MCD Application Team

Version:
V2.0.0

Date:
30-December-2016

Attention:
© COPYRIGHT(c) 2016 STMicroelectronics

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

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

Definition in file stm32f769i_discovery_sdram.h.
This file provides a set of functions needed to manage the Touch Screen on STM32F769I-DISCOVERY discovery board. More...

```c
#include "stm32f769i_discovery.h" #include "stm32f769i_discovery_ts.h"
```

Go to the source code of this file.
## Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_TS_Init</strong> (uint16_t ts_SizeX, uint16_t ts_SizeY)</td>
<td>Initializes and configures the touch screen functionalities and configures all necessary hardware resources (GPIOs, I2C, clocks..).</td>
<td></td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_TS_ITConfig</strong> (void)</td>
<td>Configures and enables the touch screen interrupts.</td>
<td></td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_TS_GetState</strong> (TS_StateTypeDef *TS_State)</td>
<td>Returns status and positions of the touch screen.</td>
<td></td>
</tr>
<tr>
<td>__weak void</td>
<td><strong>BSP_TS_INT_MspInit</strong> (void)</td>
<td>Initializes the TS_INT pin MSP.</td>
<td></td>
</tr>
</tbody>
</table>
### Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static TS_DrvTypeDef *</code></td>
<td><code>ts_driver</code></td>
<td></td>
</tr>
<tr>
<td><code>static uint8_t</code></td>
<td><code>ts_orientation</code></td>
<td></td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><code>I2C_Address</code></td>
<td><code>= 0</code></td>
</tr>
<tr>
<td><code>char *</code></td>
<td><code>ts_event_string_tab</code></td>
<td><code>[TOUCH_EVENT_NB_MAX]</code> Table for touchscreen event information display on LCD: table indexed on enum <code>TS_TouchEventTypeDef</code> information.</td>
</tr>
<tr>
<td><code>char *</code></td>
<td><code>ts_gesture_id_string_tab</code></td>
<td><code>[GEST_ID_NB_MAX]</code> Table for touchscreen gesture Id information display on LCD: table indexed on enum <code>TS_GestureIdTypeDef</code> information.</td>
</tr>
</tbody>
</table>
Detailed Description

This file provides a set of functions needed to manage the Touch Screen on STM32F769I-DISCOVERY discovery board.

Author:
MCD Application Team

Version:
V2.0.0

Date:
30-December-2016

Attention:
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

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

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

Definition in file stm32f769i_discovery_ts.c.
### stm32f769i_discovery_ts.h File Reference

This file contains the common defines and functions prototypes for the `stm32f769i_discovery_ts.c` driver. More...

```c
#include "stm32f769i_discovery.h" #include "stm32f769i_discovery_lcd.h" #include "../Components/ft6x06/ft6x06.h"
```

Go to the source code of this file.
Data Structures

struct TS_StateTypeDef
TS_StateTypeDef Define TS State structure. More...
#defines

<table>
<thead>
<tr>
<th>Define</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS_MAX_NB_TOUCH</td>
<td>((uint32_t) FT6206_MAX_DETECTABLE_TOUCH)</td>
</tr>
<tr>
<td></td>
<td>With FT6206: maximum 2 touches detected simultaneously.</td>
</tr>
<tr>
<td>TS_NO_IRQ_PENDING</td>
<td>((uint8_t) 0)</td>
</tr>
<tr>
<td>TS_IRQ_PENDING</td>
<td>((uint8_t) 1)</td>
</tr>
<tr>
<td>TS_SWAP_NONE</td>
<td>((uint8_t) 0x01)</td>
</tr>
<tr>
<td>TS_SWAP_X</td>
<td>((uint8_t) 0x02)</td>
</tr>
<tr>
<td>TS_SWAP_Y</td>
<td>((uint8_t) 0x04)</td>
</tr>
<tr>
<td>TS_SWAP_XY</td>
<td>((uint8_t) 0x08)</td>
</tr>
</tbody>
</table>
Enumerations

```c
enum TS_StatusTypeDef { TS_OK = 0x00, TS_ERROR = 0x01, TS_TIMEOUT = 0x02, TS_DEVICE_NOT_FOUND = 0x03 }

TS_StatusTypeDef Define BSP_TSPxxx() functions possible return value, when status is returned by those functions. More...
```

```c
enum TS_GestureIdTypeDef {
    GEST_ID_NO_GESTURE = 0x00,
    GEST_ID_MOVE_UP = 0x01,
    GEST_ID_MOVE_RIGHT = 0x02,
    GEST_ID_MOVE_DOWN = 0x03,
    GEST_ID_MOVE_LEFT = 0x04,
    GEST_ID_ZOOM_IN = 0x05,
    GEST_ID_ZOOM_OUT = 0x06,
    GEST_ID_NB_MAX = 0x07
}

TS_GestureIdTypeDef Define Possible managed gesture identification values returned by touchscreen driver. More...
```

```c
enum TS_TouchEventTypeDef {
    TOUCH_EVENT_NO_EVT = 0x00,
    TOUCH_EVENT_PRESS_DOWN = 0x01,
    TOUCH_EVENT_LIFT_UP = 0x02,
    TOUCH_EVENT_CONTACT = 0x03,
    TOUCH_EVENT_NB_MAX = 0x04
}

TS_TouchEventTypeDef Define Possible touch events kinds as returned values by touchscreen IC Driver. More...
```
## Functions

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td><strong>BSP_TS_Init</strong> (uint16_t ts_SizeX, uint16_t ts_SizeY)</td>
<td>Initializes and configures the touch screen functionalities and configures all necessary hardware resources (GPIOs, I2C, clocks..).</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_TS_GetState</strong> (TS_StateTypeDef *TS_State)</td>
<td>Returns status and positions of the touch screen.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_TS_ITConfig</strong> (void)</td>
<td>Configures and enables the touch screen interrupts.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_TS_INT_MspInit</strong> (void)</td>
<td>Initializes the TS_INT pin MSP.</td>
</tr>
</tbody>
</table>
### Variables

<table>
<thead>
<tr>
<th>char *</th>
<th><code>ts_event_string_tab [TOUCH_EVENT_NB_MAX]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Table for touchscreen event information display on LCD: table indexed on enum <code>TS_TouchEventTypeDef</code> information.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>char *</th>
<th><code>ts_gesture_id_string_tab [GEST_ID_NB_MAX]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Table for touchscreen gesture Id information display on LCD: table indexed on enum <code>TS_GestureIdTypeDef</code> information.</td>
</tr>
</tbody>
</table>
Detailed Description

This file contains the common defines and functions prototypes for the *stm32f769i_discovery_ts.c* driver.

**Author:**
MCD Application Team

**Version:**
V2.0.0

**Date:**
30-December-2016

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

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

Definition in file stm32f769i_discovery_ts.h.
STM32F769I-Discovery BSP User Manual

Here is a list of all modules:

- **BSP**
  - **STM32F769I_DISCOVERY**
    - **STM32F769I_DISCOVERY LOW LEVEL**
      - STM32F769I Discovery Low Level Private Typedef
      - LOW_LEVEL Private Defines
      - LOW_LEVEL Private Macros
      - LOW_LEVEL Private Variables
      - LOW_LEVEL Private FunctionPrototypes
    - **BSP Public Functions**
    - **STM32F769I_DISCOVERY_LOW_LEVEL Private Functions**
    - **STM32F769I Discovery Low Level Exported Types**
      - STM32F769I Discovery Low Level Exported Constants
      - STM32F769I Discovery Low Level Led
      - STM32F769I Discovery Low Level Button
    - **STM32F769I Discovery Low Level Exported Macros**
    - **STM32F769I Discovery Low Level Exported Functions**
    - **Exported_types**
  - **STM32F769I_DISCOVERY AUDIO**
    - **STM32F769I_DISCOVERY_AUDIO Private Types**
    - **STM32F769I_DISCOVERY_AUDIO Private Defines**
    - **STM32F769I_DISCOVERY_AUDIO Private Macros**
- STM32F769I_DISCOVERY_AUDIO Private Variables
- STM32F769I_DISCOVERY_AUDIO Private Function Prototypes
- STM32F769I_DISCOVERY_AUDIO_Out Private Functions
- STM32F769I_DISCOVERY_AUDIO_In Private Functions
- STM32F769I_DISCOVERY_AUDIO_Exported Types
- STM32F769I_DISCOVERY_AUDIO_Exported Constants
  - BSP Audio Sample Rate
- STM32F769I_DISCOVERY_AUDIO_Exported Macros
- STM32F769I_DISCOVERY_AUDIO_OUT_Exported Functions
- STM32F769I_DISCOVERY_AUDIO_IN_Exported Functions

- STM32F769I_DISCOVERY_LCD
  - LCD Private Defines
  - LCD Private TypesDefinitions
  - LCD Private Macros
  - STM32F769I_DISCOVERY_LCD_Exported Variables
  - LCD Private Variables
  - LCD Private FunctionPrototypes
  - LCD Exported Functions
  - STM32F769I_DISCOVERY_LCD_Exported Constants
  - STM32F769I_DISCOVERY_LCD_Exported Types
  - STM32F769I_DISCOVERY_LCD_Exported Macro
    - STM32F769I_DISCOVERY_LCD_Exported Variables
    - LCD Exported Functions

- STM32F769I_DISCOVERY_QSPI
  - STM32F769I_DISCOVERY_QSPI Private Variables
  - STM32F769I_DISCOVERY_QSPI Private Functions
  - STM32F769I_DISCOVERY_QSPI_Exported
Functions
- STM32F769I_DISCOVERY_QSPI Exported Constants
- STM32F769I_DISCOVERY_QSPI Exported Types
- STM32F769I_DISCOVERY_SD
  - STM32F769I Discovery Sd Private TypesDef
  - STM32F769I Discovery Sd Private Defines
  - STM32F769I Discovery Sd Private Macro
  - STM32F769I Discovery Sd Private Variables
  - STM32F769I Discovery Sd Private Prototypes
  - STM32F769I Discovery Sd Private Functions
  - SD Exported Types
  - SD Exported Constants
  - STM32F769I Discovery SD Exported Macro
  - STM32F769I Discovery SD Exported Functions
- STM32F769I_DISCOVERY_SDRAM
  - SDRAM Private Types Definitions
  - SDRAM Private Defines
  - SDRAM Private Macros
  - SDRAM Private Variables
  - SDRAM Private Function Prototypes
  - SDRAM Private Functions
  - SDRAM Exported Types
    - SDRAM Exported Constants
  - SDRAM Exported Macro
  - SDRAM Exported Functions
- STM32F769I_DISCOVERY_TS
  - TS Private Types Definitions
  - TS Private Types Defines
  - TS Private Macros
  - TS Imported Variables
  - TS Private Variables
  - TS Private Function Prototypes
  - TS Public Functions
    - TS Private Functions
  - TS Exported Constants
  - TS Exported Types
- TS Exported Functions
- STM32F769I_DISCOVERY EEPROM
  - EEPROM Private Types
  - EEPROM Private Defines
  - EEPROM Private Macros
  - EEPROM Private Variables
  - EEPROM Private Function Prototypes
  - EEPROM Private Functions
  - EEPROM Exported Types
  - EEPROM Exported Constants
  - EEPROM Exported Macros
  - EEPROM Exported Functions

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery
BSP User Manual by doxygen 1.7.6.1
Here are the data structures with brief descriptions:

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUDIOIN_TypeDef</td>
<td>LCD Drawing main properties</td>
</tr>
<tr>
<td>LCD_DrawPropTypeDef</td>
<td>LCD Drawing point (pixel) geometric definition</td>
</tr>
<tr>
<td>Point</td>
<td>LCD Drawing point (pixel) geometric definition</td>
</tr>
<tr>
<td>QSPI_Info</td>
<td></td>
</tr>
<tr>
<td>TS_StateTypeDef</td>
<td>Define TS State structure</td>
</tr>
</tbody>
</table>

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
Here is a list of all files with brief descriptions:

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>stm32f769i_discovery.c</strong></td>
<td>This file provides a set of firmware functions to manage LEDs, push-buttons, external SDRAM, QSPI Flash, RF EEPROM, available on STM32F769I-Discovery board (MB1225) from STMicroelectronics.</td>
</tr>
<tr>
<td><strong>stm32f769i_discovery.h</strong></td>
<td>This file contains definitions for STM32F769I-Discovery LEDs, push-buttons hardware resources.</td>
</tr>
<tr>
<td><strong>stm32f769i_discovery_audio.c</strong></td>
<td>This file provides the Audio driver for the STM32F769I-DISCOVERY board.</td>
</tr>
<tr>
<td><strong>stm32f769i_discovery_audio.h</strong></td>
<td>This file contains the common defines and functions prototypes for the <strong>stm32f769i_discovery_audio.c</strong> driver.</td>
</tr>
<tr>
<td><strong>stm32f769i_discovery_eeprom.c</strong></td>
<td>This file provides a set of functions needed to manage an I2C M24LR64 EEPROM memory.</td>
</tr>
<tr>
<td><strong>stm32f769i_discovery_eeprom.h</strong></td>
<td>This file contains all the functions.</td>
</tr>
</tbody>
</table>
prototypes for the firmware driver

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stm32f769i_discovery_lcd.c</td>
<td>This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM32F769I-DISCOVERY board</td>
</tr>
<tr>
<td>stm32f769i_discovery_lcd.h</td>
<td>This file contains the common defines and functions prototypes for the stm32469i_discovery_lcd driver</td>
</tr>
<tr>
<td>stm32f769i_discovery_qspi.c</td>
<td>This file includes a standard driver for the MX25L512 QSPI memory mounted on STM32F769I-DISCOVERY board</td>
</tr>
<tr>
<td>stm32f769i_discovery_qspi.h</td>
<td>This file contains the common defines and functions prototypes for the stm32f769i_discovery_qspi driver</td>
</tr>
<tr>
<td>stm32f769i_discovery_sd.c</td>
<td>This file includes the uSD card driver mounted on STM32F769I-DISCOVERY board</td>
</tr>
<tr>
<td>stm32f769i_discovery_sd.h</td>
<td>This file contains the common defines and functions prototypes for the stm32f769i_discovery_sd driver</td>
</tr>
<tr>
<td>stm32f769i_discovery_sdram.c</td>
<td>This file includes the SDRAM driver for the MT48LC4M32B2B5-6A memory device mounted on STM32F769I-DISCOVERY board</td>
</tr>
<tr>
<td>stm32f769i_discovery_sdram.h</td>
<td>This file contains the common defines and functions prototypes</td>
</tr>
<tr>
<td>File Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>stm32f769i_discovery_ts.c</td>
<td>This file provides a set of functions needed to manage the Touch Screen on STM32F769I-DISCOVERY discovery board</td>
</tr>
<tr>
<td>stm32f769i_discovery_ts.h</td>
<td>This file contains the common defines and functions prototypes for the stm32f769i_discovery_ts driver</td>
</tr>
</tbody>
</table>
# STM32F769I-Discovery BSP User Manual

## Directories

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

- **Drivers**
  - **BSP**
    - **STM32F769I-Discovery**

---

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery BSP User Manual by [doxygen](https://www.doxygen.org) 1.7.6.1
## STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**STM32F769I_DISCOVERY LOW LEVEL**

STM32F769I_DISCOVERY
## Modules

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F769I Discovery Low Level Private Type Def</td>
<td></td>
</tr>
<tr>
<td>LOW_LEVEL Private Defines</td>
<td></td>
</tr>
<tr>
<td>LOW_LEVEL Private Macros</td>
<td></td>
</tr>
<tr>
<td>LOW_LEVEL Private Variables</td>
<td></td>
</tr>
<tr>
<td>LOW_LEVEL Private Function Prototypes</td>
<td></td>
</tr>
<tr>
<td>BSP Public Functions</td>
<td></td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY_LOW_LEVEL Private Functions</td>
<td></td>
</tr>
<tr>
<td>STM32F769I Discovery Low Level Exported Types</td>
<td></td>
</tr>
</tbody>
</table>
### Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>AUDIO_IO_Write</strong> (uint8_t Addr, uint16_t Reg, uint16_t Value)</td>
<td>Writes a single data.</td>
</tr>
<tr>
<td>uint16_t</td>
<td><strong>AUDIO_IO_Read</strong> (uint8_t Addr, uint16_t Reg)</td>
<td>Reads a single data.</td>
</tr>
<tr>
<td>void</td>
<td><strong>AUDIO_IO_Delay</strong> (uint32_t Delay)</td>
<td>AUDIO Codec delay.</td>
</tr>
<tr>
<td>HAL_StatusTypeDef</td>
<td><strong>EEPROM_IO_WriteData</strong> (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize)</td>
<td>Write data to I2C EEPROM driver in using DMA channel.</td>
</tr>
<tr>
<td>HAL_StatusTypeDef</td>
<td><strong>EEPROM_IO_ReadData</strong> (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize)</td>
<td>Read data from I2C EEPROM driver in using DMA channel.</td>
</tr>
<tr>
<td>HAL_StatusTypeDef</td>
<td><strong>EEPROM_IO_IsDeviceReady</strong> (uint16_t DevAddress, uint32_t Trials)</td>
<td>Checks if target device is ready for communication.</td>
</tr>
<tr>
<td>void</td>
<td><strong>TS_IO_Init</strong> (void)</td>
<td>Initializes Touchscreen low level.</td>
</tr>
<tr>
<td>void</td>
<td><strong>TS_IO_Write</strong> (uint8_t Addr, uint8_t Reg, uint8_t Value)</td>
<td>Writes a single data.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>TS_IO_Read</strong> (uint8_t Addr, uint8_t Reg)</td>
<td>Reads a single data.</td>
</tr>
<tr>
<td>uint16_t</td>
<td><strong>TS_IO_ReadMultiple</strong> (uint8_t Addr, uint8_t Reg, uint8_t *Buffer, uint16_t Length)</td>
<td>Reads multiple data with I2C communication channel from TouchScreen.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td><code>TS_IO_WriteMultiple</code></td>
<td>Writes multiple data with I2C communication channel from MCU to TouchScreen.</td>
<td></td>
</tr>
<tr>
<td><code>TS_IO_Delay</code></td>
<td>Delay function used in TouchScreen low level driver.</td>
<td></td>
</tr>
<tr>
<td><code>OTM8009A_IO_Delay</code></td>
<td>OTM8009A delay.</td>
<td></td>
</tr>
<tr>
<td><code>HDMI_IO_Init</code></td>
<td>Initializes HDMI IO low level.</td>
<td></td>
</tr>
<tr>
<td><code>HDMI_IO_Write</code></td>
<td>HDMI writes single data.</td>
<td></td>
</tr>
<tr>
<td><code>HDMI_IO_Read</code></td>
<td>Reads single data with I2C communication channel from HDMI bridge.</td>
<td></td>
</tr>
<tr>
<td><code>HDMI_IO_Delay</code></td>
<td>HDMI delay.</td>
<td></td>
</tr>
</tbody>
</table>
Function Documentation

```c
void AUDIO_IO_Delay ( uint32_t Delay )
```

AUDIO Codec delay.

**Parameters:**
- **Delay:** Delay in ms

Definition at line 611 of file stm32f769i_discovery.c.

```c
uint16_t AUDIO_IO_Read ( uint8_t Addr, uint16_t Reg )
```

Reads a single data.

**Parameters:**
- **Addr:** I2C address
- **Reg:** Reg address

**Return values:**
- **Data** to be read

Definition at line 592 of file stm32f769i_discovery.c.

References `hi2cAudioHandler`, and `I2Cx_ReadMultiple()`.

```c
void AUDIO_IO_Write ( uint8_t Addr, uint16_t Reg, uint16_t Value )
```
Writes a single data.

**Parameters:**
- **Addr.**: I2C address
- **Reg.**: Reg address
- **Value.**: Data to be written

**Return values:**
- None

Definition at line 575 of file `stm32f769i_discovery.c`.

References **hi2cAudioHandler**, and **I2Cx_WriteMultiple()**.

```c
HAL_StatusTypeDef EEPROM_IO_IsDeviceReady ( uint16_t DevAddress,
                                          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 659 of file `stm32f769i_discovery.c`.

References **hi2cExtHandler**, and **I2Cx_IsDeviceReady()**.

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

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

Read data from I2C EEPROM driver in using DMA channel.

Parameters:
- **DevAddress**: Target device address
- **MemAddress**: Internal memory address
- **pBuffer**: Pointer to data buffer
- **BufferSize**: Amount of data to be read

Return values:
- **HAL** status

Definition at line 647 of file stm32f769i_discovery.c.

References hI2cExtHandler, and I2Cx_ReadMultiple().

Referenced by BSP_EEPROM_ReadBuffer().

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

Write data to I2C EEPROM driver in using DMA channel.

Parameters:
**DevAddress,** Target device address

**MemAddress,** Internal memory address

**pBuffer,** Pointer to data buffer

**BufferSize,** Amount of data to be sent

**Return values:**

HAL status

Definition at line 634 of file `stm32f769i_discovery.c`.

References `hi2cExtHandler`, and `I2Cx_WriteMultiple()`.

Referenced by `BSP_EEPROM_WritePage()`.

```c
void HDMI_IO_Delay ( uint32_t Delay )
```

HDMI delay.

**Parameters:**

Delay, Delay in ms

**Return values:**

None

Definition at line 793 of file `stm32f769i_discovery.c`.

Referenced by `LCD_IO_GetID()`.

```c
void HDMI_IO_Init ( void )
```

Initializes HDMI IO low level.

**Return values:**

None
Definition at line 755 of file stm32f769i_discovery.c.
 References hI2cAudioHandler, and I2Cx_Init().
 Referenced by LCD_IO_GetID().

```c
uint8_t HDMI_IO_Read ( uint8_t Addr,
                        uint8_t Reg
)
```

Reads single data with I2C communication channel from HDMI bridge.

**Parameters:**
- **Addr:** I2C address
- **Reg:** Register address

**Return values:**
- **Read** data

Definition at line 779 of file stm32f769i_discovery.c.
 References hI2cAudioHandler, and I2Cx_ReadMultiple().
 Referenced by LCD_IO_GetID().

```c
void HDMI_IO_Write ( uint8_t Addr,
                        uint8_t Reg,
                        uint8_t Value
)
```

HDMI writes single data.

**Parameters:**
- **Addr:** I2C address
Register address

Data to be written

Return values:
None

Definition at line 767 of file stm32f769i_discovery.c.

References hI2cAudioHandler, and I2Cx_WriteMultiple().

---

void OTM8009A_IO_Delay ( uint32_t Delay )

OTM8009A delay.

Parameters:
Delay: Delay in ms

Definition at line 745 of file stm32f769i_discovery.c.

---

void TS_IO_Delay ( uint32_t Delay )

Delay function used in TouchScreen low level driver.

Parameters:
Delay: Delay in ms

Return values:
None

Definition at line 735 of file stm32f769i_discovery.c.

---

void TS_IO_Init ( void )

Initializes Touchscreen low level.
Return values:

None

Definition at line 670 of file stm32f769i_discovery.c.

References hI2cAudioHandler, and I2Cx_Init().

```c
uint8_t TS_IO_Read ( uint8_t Addr,
                    uint8_t Reg )
```

Reads a single data.

**Parameters:**

- **Addr:** I2C address
- **Reg:** Reg address

**Return values:**

- **Data** to be read

Definition at line 693 of file stm32f769i_discovery.c.

References hI2cAudioHandler, and I2Cx_ReadMultiple().

```c
uint16_t TS_IO_ReadMultiple ( uint8_t Addr,
                               uint8_t Reg,
                               uint8_t * Buffer,
                               uint16_t Length )
```

Reads multiple data with I2C communication channel from TouchScreen.

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

**Return values:**
Number of read data

Definition at line 711 of file stm32f769i_discovery.c.

References hl2cAudioHandler, and I2Cx_ReadMultiple().

```c
void TS_IO_Write ( uint8_t Addr,
                 uint8_t Reg,
                 uint8_t Value
)
```

Writes a single data.

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

**Return values:**
None

Definition at line 682 of file stm32f769i_discovery.c.

References hl2cAudioHandler, and I2Cx_WriteMultiple().

```c
void TS_IO_WriteMultiple ( uint8_t Addr,
                          uint8_t Reg,
                          uint8_t * Buffer,
```
uint16_t Length
}

Writes multiple data with I2C communication channel from MCU to TouchScreen.

**Parameters:**
- **Addr:** I2C address
- **Reg:** Register address
- **Buffer:** Pointer to data buffer
- **Length:** Length of the data

**Return values:**
- None

Definition at line 725 of file `stm32f769i_discovery.c`.

References `hI2cAudioHandler`, and `I2Cx_WriteMultiple()`.

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
## STM32F769I Discovery

### Low Level Exported Types

**STM32F769I_DISCOVERY LOW LEVEL**
# Modules

<table>
<thead>
<tr>
<th>Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F769I Discovery Low Level Exported Constants</td>
</tr>
<tr>
<td>STM32F769I Discovery Low Level Exported Macros</td>
</tr>
<tr>
<td>STM32F769I Discovery Low Level Exported Functions</td>
</tr>
<tr>
<td>Exported_types</td>
</tr>
</tbody>
</table>
Defines

#define BUTTON_USER BUTTON_WAKEUP
## Enumerations

<table>
<thead>
<tr>
<th>Enum Type</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Led_TypeDef</td>
<td><code>Led_TypeDef { LED1 = 0, LED_RED = LED1, LED2 = 1, LED_GREEN = LED2 }</code></td>
<td>Define for STM32F769I_DISCOVERY board. More...</td>
</tr>
<tr>
<td>Button_TypeDef</td>
<td><code>Button_TypeDef { BUTTON_WAKEUP = 0 }</code></td>
<td>Button_TypeDef STM32F769I_DISCOVERY board Buttons definitions. More...</td>
</tr>
<tr>
<td>ButtonMode_TypeDef</td>
<td><code>ButtonMode_TypeDef { BUTTON_MODE_GPIO = 0, BUTTON_MODE_EXTI = 1 }</code></td>
<td>ButtonMode_TypeDef STM32F769I_DISCOVERY board Buttons Modes definitions. More...</td>
</tr>
</tbody>
</table>
Define Documentation

`#define BUTTON_USER BUTTON_WAKEUP`

Definition at line 93 of file `stm32f769i_discovery.h`. 
Enumeration Type Documentation

**enum** Button_TypeDef

Button_TypeDef STM32F769I_DISCOVERY board Buttons definitions.

**Enumerator:**

    BUTTON_WAKEUP

Definition at line 88 of file stm32f769i_discovery.h.

**enum** ButtonMode_TypeDef

ButtonMode_TypeDef STM32F769I_DISCOVERY board Buttons Modes definitions.

**Enumerator:**

    BUTTON_MODE_GPIO
    BUTTON_MODE_EXTI

Definition at line 98 of file stm32f769i_discovery.h.

**enum** Led_TypeDef

Define for STM32F769I_DISCOVERY board.

Led_TypeDef STM32F769I_DISCOVERY board leds definitions.

**Enumerator:**

    LED1
    LED_RED
    LED2
LED_GREEN

Definition at line 77 of file stm32f769i_discovery.h.
## Data Structure Index

<table>
<thead>
<tr>
<th>A</th>
<th>L</th>
<th>P</th>
<th>Q</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUDIOIN_TypeDef</td>
<td>LCD_DrawPropTypeDef</td>
<td>Point</td>
<td>QSPI_Info</td>
<td></td>
</tr>
</tbody>
</table>

Generated on Fri Dec 30 2016 18:30:08 for STM32F769I-Discovery BSP User Manual by [doxygen](https://www.doxygen.org/) 1.7.6.1
# STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td><strong>Data Structures</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**STM32F769I_DISCOVERY_AUDIO**

**Private Types**

STM32F769I_DISCOVERY_AUDIO
Data Structures

struct AUDIOIN_TypeDef

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
stm32f769i_discovery_audio.c

Go to the documentation of this file.

```c
/**
 * @file    stm32f769i_discovery_audio.c
 * @author  MCD Application Team
 * @version V2.0.0
 * @date    30-December-2016
 * @brief   This file provides the Audio driver for the STM32F769I-DISCOVERY
 *          board.
 * @verbatim
 * How To use this driver:
 * + This driver supports STM32F7xx devices on STM32F769I-DISCOVERY (MB1225) Evaluation boards.
 * + Call the function BSP_AUDIO_OUT_Init(OutputDevice:
 *   physical output mode (OUTPUT_DEVICE_SPEAKER,
 *   OUTPUT_DEVICE_HEADPHONE or OUTPUT_DEVICE_BOT
```

```
Volume
: Initial volume to be set (0 is min (mute), 1 is max (100%))

AudioFreq
: Audio frequency in Hz (8000, 16000, 22500, 32000...)

this parameter is relative to the audio file/stream type.

This function configures all the hardware required for the audio application (codec, I2C, SAI, GPIOs, DMA and interrupt if needed). This function returns AUDIO_OK if configuration is OK.

If the returned value is different from AUDIO_OK or the function is stuck then the communication with the codec has failed (try to un-plug the power or reset device in this case).

- OUTPUT_DEVICE_SPEAKER : only speaker will be set as output for the audio stream.
- OUTPUT_DEVICE/headphone : only headphones will be set as output for the audio stream.
- OUTPUT_DEVICE_BOTH : both Speaker and Headphone are used as outputs for the audio stream at the same time.

Note. On STM32F769I-DISCOVERY SAI_DMA is configured in CIRCULAR mode. Due to this the application does NOT need to call BSP_AUDIO_OUT_ChangeBuffer() to assure streaming.

+ Call the function BSP_AUDIO_OUT_Play(
pBuffer: pointer to the audio data file address
Size: size of the buffer to be sent in Bytes
to start playing (for the first time) from the audio file/stream.
+ Call the function BSP_AUDIO_OUT_Pause() to pause playing
+ Call the function BSP_AUDIO_OUT_Resume() to resume playing.
Note. After calling BSP_AUDIO_OUT_Pause() function for pause, only BSP_AUDIO_OUT_Resume() should be called for resume (it is not allowed to call BSP_AUDIO_OUT_Play() in this case).
Note. This function should be called only when the audio file is played or paused (not stopped).
+ For each mode, you may need to implement the relative callback functions into your code.
The Callback functions are named BSP_AUDIO_OUT_XXX_CallBack() and only their prototypes are declared in the stm32f769i_discovery_audio.h file. (refer to the example for more details on the callbacks implementations)
+ To Stop playing, to modify the volume level, the frequency, the audio frame slot, the device output mode the mute or the stop, use the functions: BSP_AUDIO_OUT_SetVolume(), AUDIO_OUT_SetFrequency(), BSP_AUDIO_OUT_SetAudioFrameSlot(), BSP_AUDIO_OUT_SetOutputMode(), BSP_AUDIO_OUT_SetMute() and BSP_AUDIO_OUT_Stop().
Call the function BSP_AUDIO_IN_Init(
AudioFreq: Audio frequency in Hz (8000, 16000, 22500, 32000...)
this parameter is relative to the audio file/stream type.
BitRes: Bit resolution fixed to 16bit
ChnlNbr: Number of channel to be configured for the DFSDM peripheral

This function configures all the hardware required for the audio in application (DFSDM filters and channels,
Clock source for DFSDM peripheral, GPI Os, DMA and interrupt if needed).
This function returns AUDIO_OK if configuration is OK. If the returned value is different from AUDIO_OK then
the configuration should be wrong.

Note: On STM32F769I-DISCOVERY, four DFSDM Channel/Filters are configured and their DMA streams are configured in CIRCULAR mode.

Call the function BSP_AUDIO_IN_AllocScratch(
pScratch: pointer to scratch tables
size : size of scratch buffer)
This function must be called before BSP_AUDIO_IN_RECORD() to allocate buffer scratch for each DFSDM channel and its size.
Note: These buffers scratch are used as intermediate buffers to collect data within final record buffer.

size is the total size of the four buffers scratch; If size is 512 then the size of each is 128.

This function must be called after BSP_AUDIO_IN_Init()
+ Call the function BSP_AUDIO_IN_RECORD(
pBuf: pointer to the recorded audio data file address
Size: size of the buffer to be written in Bytes
)

to start recording from microphones.

+ Call the function BSP_AUDIO_IN_Pause() to pause recording
+ Call the function BSP_AUDIO_IN_Resume() to recording playing.

Note. After calling BSP_AUDIO_IN_Pause() function for pause, only BSP_AUDIO_IN_Resume() should be called for resume (it is not allowed to call BSP_AUDIO_IN_RECORD() in this case).

+ Call the function BSP_AUDIO_IN_Stop() to stop recording
+ For each mode, you may need to implement the relative callback functions into your code.

The Callback functions are named BSP_AUDIO_IN_XXX_CallBack() and only their prototypes are declared in the stm32f769i_discovery_audio.h file. (refer to the example for more details on the callbacks implementations)

Driver architecture:
This driver provides the High Audio Layer: consists of the function API exported in the stm32f769i_discovery_audio.h file (BSP_AUDIO_OUT_Init(), BSP_AUDIO_OUT_Play() ...)

This driver provide also the Media Access Layer (MAL): which consists of functions allowing to access the media containing/ providing the audio file/stream. These functions are also included as local functions into the stm32f769i_discovery_audio.c file (DFSDMx_Init(), DFSDMx_DeInit(), SAIx_Init() and SAIx_DeInit())

Known Limitations:

1- If the TDM Format used to play in parallel 2 audio Stream (the first Stream is configured in codec SLOT0 and second Stream in SLOT1) the Pause/Resume, volume and mute feature will control the both streams.

2- Parsing of audio file is not implemented (in order to determine audio file properties: Mono/Stereo, Data size, File size, Audio Frequency, Audio Data header size ...). The configuration is fixed for the given audio file.

3- Supports only Stereo audio streaming.

4- Supports only 16-bits audio data size.

@endverbatim
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer,
3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
OR BUSINESS INTERRUPTION) HOWEVER

* CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
* OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

*/

/**
 @addtogroup BSP
 @{
*/

/**
 @addtogroup STM32F769I_DISCOVERY
 @{
*/

/**
 @defgroup STM32F769I_DISCOVERY_AUDIO	STM32F769I_DISCOVERY	AUDIO
 @brief	This file includes the low layer driver for wm8994 Audio Codec
 available on STM32F769I-DISCOVERY discoveryuation board(MB1225).
*/

/**
 @defgroup STM32F769I_DISCOVERY_AUDIO_Private_Types	STM32F769I_DISCOVERY_AUDIO	Private Types
*/
typedef struct {
    uint16_t *pRecBuf; /* Pointer to record user buffer */
    uint32_t RecSize; /* Size to record in mono, double size to record in stereo */
} AUDIOIN_TypeDef;

/** @defgroup STM32F769I_DISCOVERY_AUDIO_Private_Defines STM32F769I_DISCOVERY_AUDIO Private Defines */
*/

/** @defgroup STM32F769I_DISCOVERY_AUDIO_Private_Macros STM32F769I_DISCOVERY_AUDIO Private Macros */
*/

/*### RECORD ###*/
#define DFSDM_OVER_SAMPLING(__FREQUENCY__) \  
    (__FREQUENCY__ == AUDIO_FREQUENCY_8K) ? 256 \  
    : (__FREQUENCY__ == AUDIO_FREQUENCY_11K) ? 256 \  
    : (__FREQUENCY__ == AUDIO_FREQUENCY_16K) ? 128 \  
    : (__FREQUENCY__ == AUDIO_FREQUENCY_22K) ? 128 \  
    : (__FREQUENCY__ == AUDIO_FREQUENCY_32K) ? 128 \  
    : 128 \  
    : __FREQUENCY__; /*### RECORD ###*/
#define DFSDM_CLOCK_DIVIDER(__FREQUENCY__)  
    ((__FREQUENCY__ == AUDIO_FREQUENCY_8K) ? 24 \ 
    ((__FREQUENCY__ == AUDIO_FREQUENCY_11K) ? 4 \ 
    ((__FREQUENCY__ == AUDIO_FREQUENCY_16K) ? 4 \ 
    ((__FREQUENCY__ == AUDIO_FREQUENCY_22K) ? 4 \ 
    ((__FREQUENCY__ == AUDIO_FREQUENCY_32K) ? 24 \ 
    ((__FREQUENCY__ == AUDIO_FREQUENCY_44K) ? 4 \ 
    ((__FREQUENCY__ == AUDIO_FREQUENCY_48K) ? 25 : 25 \ 

#define DFSDM_FILTER_ORDER(__FREQUENCY__)  
    ((__FREQUENCY__ == AUDIO_FREQUENCY_8K) ? DFSDM_FILTER_SINC3_ORDER \ 
    ((__FREQUENCY__ == AUDIO_FREQUENCY_11K) ? DFSDM_FILTER_SINC3_ORDER \ 
    ((__FREQUENCY__ == AUDIO_FREQUENCY_16K) ? DFSDM_FILTER_SINC3_ORDER \ 
    ((__FREQUENCY__ == AUDIO_FREQUENCY_22K) ? DFSDM_FILTER_SINC3_ORDER \ 
    ((__FREQUENCY__ == AUDIO_FREQUENCY_32K) ? DFSDM_FILTER_SINC4_ORDER \ 
    ((__FREQUENCY__ == AUDIO_FREQUENCY_44K) ? DFSDM_FILTER_SINC3_ORDER \ 
    ((__FREQUENCY__ == AUDIO_FREQUENCY_48K) ? DFSDM_FILTER_SINC3_ORDER : DFSDM_FILTER_SINC5
#define DFSDM_RIGHT_BIT_SHIFT(__FREQUENCY__) 

 (__FREQUENCY__ == AUDIO_FREQUENCY_8K) ? 8 
: ((__FREQUENCY__ == AUDIO_FREQUENCY_11K) ? 8 
: ((__FREQUENCY__ == AUDIO_FREQUENCY_16K) ? 3 
: ((__FREQUENCY__ == AUDIO_FREQUENCY_22K) ? 4 
: ((__FREQUENCY__ == AUDIO_FREQUENCY_32K) ? 7 
: ((__FREQUENCY__ == AUDIO_FREQUENCY_44K) ? 0 
: ((__FREQUENCY__ == AUDIO_FREQUENCY_48K) ? 0 : 4)))))

/* Saturate the record PCM sample */
#define SaturaLH(N, L, H) (((N)<(L))? (L):(((N)>(H))? (H):(N)))

/**
  * @}
*/

/**
 @defgroup STM32F769I_DISCOVERY_AUDIO_Private Variables
 STM32F769I_DISCOVERY_AUDIO Private Variables
  */

/* PLAY */

AUDIO_DrvTypeDef *audio_drv;

SAI_HandleTypeDef haudio_out_sai;

SAI_HandleTypeDef haudio_in_sai;
/* RECORD */

AUDIOIN_TypeDef hAudioIn;

DFSDM_Channel_HandleTypeDef hAudioInTopLeftChannel;
DFSDM_Channel_HandleTypeDef hAudioInTopRightChannel;
DFSDM_Filter_HandleTypeDef hAudioInTopLeftFilter;
DFSDM_Filter_HandleTypeDef hAudioInTopRightFilter;
DMA_HandleTypeDef hDmaTopLeft;
DMA_HandleTypeDef hDmaTopRight;

DFSDM_Channel_HandleTypeDef hAudioInBottomLeftChannel;
DFSDM_Channel_HandleTypeDef hAudioInBottomRightChannel;
DFSDM_Filter_HandleTypeDef hAudioInBottomLeftFilter;
DFSDM_Filter_HandleTypeDef hAudioInBottomRightFilter;
DMA_HandleTypeDef hDmaBottomLeft;
DMA_HandleTypeDef hDmaBottomRight;

/* Buffers for right and left samples */

static int32_t *pScratchBuff [2*DEFAULT_AUDIO_IN_CHANNEL_NBR];
static __IO int32_t ScratchSize;

/* Channel number to be used: 2 channels by default */
static uint8_t AudioIn_ChannelNumber = DEFAULT_AUDIO_IN_CHANNEL_NBR;
/* Input device to be used: digital microphones by default */
static uint16_t AudioIn_Device = INPUTDEVICE_DIGITAL_MIC;

/* Buffers status flags */
static uint32_t DmaTopLeftRecHalfCplt = 0;
static uint32_t DmaTopLeftRecCplt = 0;
static uint32_t DmaTopRightRecHalfCplt = 0;
static uint32_t DmaTopRightRecCplt = 0;
static uint32_t DmaBottomLeftRecHalfCplt = 0;
static uint32_t DmaBottomLeftRecCplt = 0;
static uint32_t DmaBottomRightRecHalfCplt = 0;
static uint32_t DmaBottomRightRecCplt = 0;

/* Application Buffer Trigger */
static __IO uint32_t AppBuffTrigger = 0;
static __IO uint32_t AppBuffHalf = 0;

/** @defgroup STM32F769I_DISCOVERY_AUDIO_Private_Function_Prototypes STM32F769I_DISCOVERY_AUDIO_Private_Function_Prototypes */
static void SAIx_Out_Init(uint32_t AudioFreq);
static void SAIx_Out_DeInit(void);
static void SAI_AUDIO_IN_MspInit(SAI_HandleTypeDef *hsai, void *Params);
static void SAI_AUDIO_IN_MspDeInit(SAI_HandleTypeDef *hsai, void *Params);
static void SAIx_In_Init(uint32_t AudioFreq);
static void SAIx_In_DeInit(void);
static void DFSDMx_ChannelMspInit(void);
static void DFSDMx_FilterMspInit(void);
static void DFSDMx_ChannelMspDeInit(void);
static void DFSDMx_FilterMspDeInit(void);
static uint8_t DFSDMx_Init(uint32_t AudioFreq);
static uint8_t DFSDMx_DeInit(void);

/**
 * @defgroup STM32F769I_DISCOVERY_AUDIO_out_Private_Functions
 * STM32F769I_DISCOVERY_AUDIO_OutPrivate Functions
 *
 * @brief Configures the audio peripherals.
 */

* @param OutputDevice: OUTPUT_DEVICE_SPEAKER, OUTPUT_DEVICE_HEADPHONE, OUTPUT_DEVICE_BOTH.
uint8_t BSP_AUDIO_OUT_Init(uint16_t OutputDevice, uint8_t Volume, uint32_t AudioFreq)
{
    uint8_t ret = AUDIO_ERROR;
    uint32_t deviceid = 0x00;

    /* Disable SAI */
    SAIx_Out_DeInit();

    /* PLL clock is set depending by the Audio Freq (44.1khz vs 48khz groups) */
    BSP_AUDIO_OUT_ClockConfig(&haudio_out_sai, AudioFreq, NULL);

    /* SAI data transfer preparation:
    Prepare the Media to be used for the audio transfer from memory to SAI peripheral */
    haudio_out_sai.Instance = AUDIO_OUT_SAIx;
    if(HAL_SAI_GetState(&haudio_out_sai) == HAL_SAI_STATE_RESET)
    {
        /* Init the SAI MSP: this __weak function can be redefined by the application*/
        BSP_AUDIO_OUT_MspInit(&haudio_out_sai, NULL);
    }
    SAIx_Out_Init(AudioFreq);

    /* wm8994 codec initialization */
    deviceid = wm8994_drv.ReadID(AUDIO_I2C_ADD
if((deviceid) == WM8994_ID) {
    /* Reset the Codec Registers */
    wm8994_drv.Reset(AUDIO_I2C_ADDRESS);
    /* Initialize the audio driver structure */
    audio_drv = &wm8994_drv;
    ret = AUDIO_OK;
} else {
    ret = AUDIO_ERROR;
}
if(ret == AUDIO_OK) {
    /* Initialize the codec internal registers */
    audio_drv->Init(AUDIO_I2C_ADDRESS, OutputDevice, Volume, AudioFreq);
}
return ret;

/**
 * @brief Starts playing audio stream from a data buffer for a determined size.
 * @param pBuffer: Pointer to the buffer
 * @param Size: Number of audio data BYTES.
 *
 * @retval AUDIO_OK if correct communication, else wrong communication
 */
uint8_t BSP_AUDIO_OUT_Play(uint16_t*pBuffer
/* Call the audio Codec Play function */
if((audio_drv->Play(AUDIO_I2C_ADDRESS, (uint16_t*)pBuffer, Size) != 0)
{
    return AUDIO_ERROR;
}
else
{
    /* Update the Media layer and enable it for play */
    HAL_SAI_Transmit_DMA(&haudio_out_sai, (uint8_t*)pBuffer, DMA_MAX(Size / AUDIODATA_SIZE));
    return AUDIO_OK;
}

/**
 * @brief Sends n-Bytes on the SAI interface.
 * @param pData: pointer on data address
 * @param Size: number of data to be written
 * @retval None
 */
void BSP_AUDIO_OUT_ChangeBuffer(uint16_t*pData, uint16_t Size)
{
    HAL_SAI_Transmit_DMA(&haudio_out_sai, (uint8_t*)pData, Size);
}

/**
 * @brief This function Pauses the audio file stream. In case
of using DMA, the DMA Pause feature is used.

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

@retval AUDIO_OK if correct communication, else wrong communication

uint8_t BSP_AUDIO_OUT_Pause(void)
{
    /* Call the Audio Codec Pause/Resume function */
    if(audio_drv->Pause(AUDIO_I2C_ADDRESS) != 0)
    {
        return AUDIO_ERROR;
    }
    else
    {
        /* Call the Media layer pause function */
        HAL_SAI_DMAPause(&haudio_out_sai);
        /* Return AUDIO_OK when all operations are correctly done */
        return AUDIO_OK;
    }
}

/**
 * @brief Resumes the audio file stream.
 *
 * @note When calling BSP_AUDIO_OUT_Pause

...
e() function for pause, only
00402  *  BSP_AUDIO_OUT_Resume() function
00403  *  should be called for resume (use of BSP_AUDIO_OUT
00404  *  _Play()  
00405  *  function for resume could lead
to unexpected behaviour).
00404  *  @retval AUDIO_OK if correct communicatio
00405  */
00406 uint8_t BSP_AUDIO_OUT_Resume(void)
00407 {
00408   /* Call the Audio Codec Pause/Resume funct
00409       ion */
00409     if(audio_drv->Resume(AUDIO_I2C_ADDRESS) !=
00410     0)
00410     {
00411       return AUDIO_ERROR;
00412     }
00413 else
00414     {
00415 /* Call the Media layer pause/resume funct
00416       ion */
00416     HAL_SAI_DMAResume(&haudio_out_sai);
00417
00418 /* Return AUDIO_OK when all operations a
00419        re correctly done */
00419     return AUDIO_OK;
00420   }
00421 }
00422/**
00423  * @brief  Stops audio playing and Power do
00424     wn the Audio Codec.
00425  *  @param  Option: could be one of the foll
00426     owing parameters
00426  *       - CODEC_PDWN_SW: for software
00427     power off (by writing registers).
Then no need to reconfigure the Codec after power on.
- CODEC_PDWN_HW: completely shut down the codec (physically).

Then need to reconfigure the Codec after power on.

@retval AUDIO_OK if correct communication, else wrong communication

*/

uint8_t BSP_AUDIO_OUT_Stop(uint32_t Option) {
    /* Call the Media layer stop function */
    HAL_SAI_DMAStop(&haudio_out_sai);

    /* Call Audio Codec Stop function */
    if (audio_drv->Stop(AUDIO_I2C_ADDRESS, Option) != 0) {
        return AUDIO_ERROR;
    }
    else {
        if (Option == CODEC_PDWN_HW) {
            /* Wait at least 100us */
            HAL_Delay(1);
        }
        /* Return AUDIO_OK when all operations are correctly done */
        return AUDIO_OK;
    }
}

/**
 * @brief Controls the current audio volume level.
 * @param Volume: Volume level to be set i
n percentage from 0% to 100% (0 for Mute and 100 for Max volume level).

* @retval AUDIO_OK if correct communication, else wrong communication

*/

uint8_t BSP_AUDIO_OUT_SetVolume(uint8_t Volume)
{
    /* Call the codec volume control function with converted volume value */
    if (audio_drv->SetVolume(AUDIO_I2C_ADDRESS, Volume) != 0)
    {
        return AUDIO_ERROR;
    }
    else
    {
        /* Return AUDIO_OK when all operations are correctly done */
        return AUDIO_OK;
    }
}

/**
 * @brief Enables or disables the MUTE mode by software
 * @param Cmd: Could be AUDIO_MUTE_ON to mute sound or AUDIO_MUTE_OFF to unmute the codec and restore previous volume level.
 * @retval AUDIO_OK if correct communication, else wrong communication
 */

uint8_t BSP_AUDIO_OUT_SetMute(uint32_t Cmd)
{
    /* Call the Codec Mute function */
if(audio_drv->SetMute(AUDIO_I2C_ADDRESS, Cmd) != 0)
{
    return AUDIO_ERROR;
}
else
{
    /* Return AUDIO_OK when all operations are correctly done */
    return AUDIO_OK;
}

/**
 * @brief Switch dynamically (while audio file is played) the output target (speaker or headphone).
 * @param Output: The audio output target: OUTPUT_DEVICE_SPEAKER, OUTPUT_DEVICE_HEADPHONE or OUTPUT_DEVICE_BOTH
 * @retval AUDIO_OK if correct communication, else wrong communication
 */
uint8_t BSP_AUDIO_OUT_SetOutputMode(uint8_t Output)
{
    /* Call the Codec output device function */

    if(audio_drv->SetOutputMode(AUDIO_I2C_ADDRESS, Output) != 0)
    {
        return AUDIO_ERROR;
    }
    else
    {
        /* Return AUDIO_OK when all operations are correctly done */
        return AUDIO_OK;
    }
return AUDIO_OK;

/**
 * Updates the audio frequency.
 * @param AudioFreq: Audio frequency used to play the audio stream.
 * @note This API should be called after the BSP_AUDIO_OUT_Init() to adjust the audio frequency.
 * @retval None
 */

void BSP_AUDIO_OUT_SetFrequency(uint32_t AudioFreq)
{
    /* PLL clock is set depending by the Audio Freq (44.1khz vs 48khz groups) */
    BSP_AUDIO_OUT_ClockConfig(&haudio_out_sai, AudioFreq, NULL);

    /* Disable SAI peripheral to allow access to SAI internal registers */
    __HAL_SAI_DISABLE(&haudio_out_sai);

    /* Update the SAI audio frequency configuration */
    haudio_out_sai.Init.AudioFrequency = AudioFreq;
    HAL_SAI_Init(&haudio_out_sai);

    /* Enable SAI peripheral to generate MCLK */
    __HAL_SAI_ENABLE(&haudio_out_sai);
}
```c
/**
 * @brief Updates the Audio frame slot configuration.
 * @param AudioFrameSlot: specifies the audio Frame slot
 * @note This API should be called after the BSP_AUDIO_OUT_Init() to adjust the
 * audio frame slot.
 * @retval None
 */

void BSP_AUDIO_OUT_SetAudioFrameSlot(uint32_t AudioFrameSlot)
{
    /* Disable SAI peripheral to allow access to SAI internal registers */
    __HAL_SAI_DISABLE(&haudio_out_sai);

    /* Update the SAI audio frame slot configuration */
    haudio_out_sai.SlotInit.SlotActive = AudioFrameSlot;
    HAL_SAI_Init(&haudio_out_sai);

    /* Enable SAI peripheral to generate MCLK */
    __HAL_SAI_ENABLE(&haudio_out_sai);
}

/**
 * @brief De-initializes the audio out peripheral.
 * @retval None
 */

void BSP_AUDIO_OUT_DeInit(void)
{
    SAIx_Out_DeInit();

    /* DeInit the SAI MSP : this __weak function */
```
on can be rewritten by the application */

00566      BSP_AUDIO_OUT_MspDeInit(&haudio_out_sai, NULL);
00567    }
00568
00569    /**<
00570    * @brief Tx Transfer completed callbacks.
00571    * @param  hsai: SAI handle
00572    * @retval None
00573    */
00574    void HAL_SAI_TxCpltCallback(SAI_HandleTypeDef *hsai)
00575    {
00576        /* Manage the remaining file size and new address offset: This function
00577            should be coded by user (its prototype is already declared in stm32f769i_discovery_audio.
00578            h) */
00579        BSP_AUDIO_OUT_TransferComplete_CallBack();
00580    }

00581    /**<
00582    * @brief Tx Half Transfer completed callb
00583    * @param  hsai: SAI handle
00584    * @retval None
00585    */
00586    void HAL_SAI_TxHalfCpltCallback(SAI_HandleTypeDef *hsai)
00587    {
00588        /* Manage the remaining file size and new address offset: This function
00589            should be coded by user (its prototype is already declared in stm32f769i_discovery_audio.
00590            h) */
00591        BSP_AUDIO_OUT_HalfTransfer_CallBack();
00592    }
/**
 * @brief SAI error callbacks.
 * @param hsai: SAI handle
 * @retval None
 */

void HAL_SAI_ErrorCallback(SAI_HandleTypeDef *hsai)
{
    if(hsai->Instance == AUDIO_OUT_SAIx)
    {
        BSP_AUDIO_OUT_Error_CallBack();
    }
    else
    {
        BSP_AUDIO_IN_Error_CallBack();
    }
}

/**
 * @brief Manages the DMA full Transfer complete event.
 * @retval None
 */

__weak void BSP_AUDIO_OUT_TransferComplete_CallBack(void)
{
}

/**
 * @brief Manages the DMA Half Transfer complete event.
 * @retval None
 */

__weak void BSP_AUDIO_OUT_HalfTransfer_CallBack(void)
{
}
/**
 * @brief Manages the DMA FIFO error event.
 * @retval None
 */
__weak void BSP_AUDIO_OUT_Error_CallBack(void)
{
}

/**
 * @brief Initializes BSP_AUDIO_OUT MSP.
 * @param sai: SAI handle
 * @param Params
 * @retval None
 */
__weak void BSP_AUDIO_OUT_MspInit(SAI_HandleTypeDef *hsai, void *Params)
{
    static DMA_HandleTypeDef hdma_sai_tx;
    GPIO_InitTypeDef gpio_init_structure;

    /* Enable SAI clock */
    AUDIO_OUT_SAIx_CLK_ENABLE();

    /* Enable GPIO clock */
    AUDIO_OUT_SAIx_MCLK_ENABLE();
    AUDIO_OUT_SAIx_SD_FS_CLK_ENABLE();

    /* CODEC_SAI pins configuration: FS, SCK, MCK and SD pins */
    gpio_init_structure.Pin = AUDIO_OUT_SAIx_FS_PIN | AUDIO_OUT_SAIx_SCK_PIN | AUDIO_OUT_SAIx_SD_PIN;
gpio_init_structure.Mode = GPIO_MODE_AF_PP;
gpio_init_structure.Pull = GPIO_NOPULL;
gpio_init_structure.Speed = GPIO_SPEED_HIGH;
gpio_init_structure.Alternate = AUDIO_OUT_SAIx_AF;
HAL_GPIO_Init(AUDIO_OUT_SAIx_SD_FS_SCK_GPIO_PORT, &gpio_init_structure);

HAL_GPIO_Init(AUDIO_OUT_SAIx_MCLK_GPIO_PORT, &gpio_init_structure);

/* Enable the DMA clock */
AUDIO_OUT_SAIx_DMAx_CLK_ENABLE();

if(hsai->Instance == AUDIO_OUT_SAIx)
{
  /* Configure the hdma_saiTx handle parameters */
  hdma_sai_tx.Init.Channel = AUDIO_OUT_SAIx_DMAx_CHANNEL;
  hdma_sai_tx.Init.Direction = DMA_MEMORY_TO_PERIPH;
  hdma_sai_tx.Init.PeriphInc = DMA_PINC_DISABLE;
  hdma_sai_tx.Init.MemInc = DMA_MINC_ENABLE;
  hdma_sai_tx.Init.PeriphDataAlignment = AUDIO_OUT_SAIx_DMAx_PERIPH_DATA_SIZE;
  hdma_sai_tx.Init.MemDataAlignment = AUDIO_OUT_SAIx_DMAx_MEM_DATA_SIZE;
  hdma_sai_tx.Init.Mode = DMA_CIRCULAR;
  hdma_sai_tx.Init.Priority = DMA_MINC_ENABLE;
00678    hdma_sai_tx.Init.FIFOMode    = D
00679    hdma_sai_tx.Init.FIFOThreshold  = D
00680    hdma_sai_tx.Init.MemBurst      = D
00681    hdma_sai_tx.Init.PeriphBurst  = D
00682
00683    hdma_sai_tx.Instance = AUDIO_OUT_SAIx_DMAx_STREAM;
00684
00685    /* Associate the DMA handle */
00686    __HAL_LINKDMA(hsai, hdmatx, hdma_sai_tx);
00687
00688    /* Deinitialize the Stream for new transfer */
00689    HAL_DMA_DeInit(&hdma_sai_tx);
00690
00691    /* Configure the DMA Stream */
00692    HAL_DMA_Init(&hdma_sai_tx);
00693  }
00694
00695    /* SAI DMA IRQ Channel configuration */
00696    HAL_NVIC_SetPriority(AUDIO_OUT_SAIx_DMAx_IRQ, AUDIO_OUT_IRQ_PREPRIO, 0);
00697    HAL_NVIC_EnableIRQ(AUDIO_OUT_SAIx_DMAx_IRQ);
00698  }
00699
00700  /**
00701    * @brief Initializes SAI Audio IN MSP.
00702    * @param hsai: SAI handle
00703    * @param Params
00704    * @retval None
static void SAI_AUDIO_IN_MspInit(SAI_HandleTypeDef *hsai, void *Params) {
    static DMA_HandleTypeDef hdma_sai_rx;
    GPIO_InitTypeDef gpio_init_structure;

    /* Enable SAI clock */
    AUDIO_IN_SAIx_CLK_ENABLE();

    /* Enable SD GPIO clock */
    AUDIO_IN_SAIx_SD_ENABLE();

    /* CODEC_SAI pin configuration: SD pin */
    gpio_init_structure.Pin = AUDIO_IN_SAIx_SD_PIN;
    gpio_init_structure.Mode = GPIO_MODE_AF_PP;
    gpio_init_structure.Pull = GPIO_NOPULL;
    gpio_init_structure.Speed = GPIO_SPEED_FAST;
    gpio_init_structure.Alternate = AUDIO_IN_SAIx_AF;
    HAL_GPIO_Init(AUDIO_IN_SAIx_SD_GPIO_PORT, &gpio_init_structure);

    /* Enable Audio INT GPIO clock */
    AUDIO_IN_INT_GPIO_ENABLE();

    /* Audio INT pin configuration: input */
    gpio_init_structure.Pin = AUDIO_IN_INT_GPIO_PIN;
    gpio_init_structure.Mode = GPIO_MODE_INPUT;
    gpio_init_structure.Pull = GPIO_NOPULL;
    gpio_init_structure.Speed = GPIO_SPEED_FAST;
    HAL_GPIO_Init(AUDIO_IN_INT_GPIO_PORT, &gpio_init_structure);
/* Enable the DMA clock */
AUDIO_IN_SAIx_DMAx_CLK_ENABLE();

if (hsai->Instance == AUDIO_IN_SAIx) {

    /* Configure the hdma_sai_rx handle parameters */
    hdma_sai_rx.Init.Channel = AUDIO_IN_SAIx_DMAx_CHANNEL;
    hdma_sai_rx.Init.Direction = DMA_PERIPH_TO_MEMORY;
    hdma_sai_rx.Init.PeriphInc = DMA_PINC_DISABLE;
    hdma_sai_rx.Init.MemInc = DMA_MINC_ENABLE;
    hdma_sai_rx.Init.PeriphDataAlignment = AUDIO_IN_SAIx_DMAx_PERIPH_DATA_SIZE;
    hdma_sai_rx.Init.MemDataAlignment = AUDIO_IN_SAIx_DMAx_MEM_DATA_SIZE;
    hdma_sai_rx.Init.Mode = DMA_CIRCULAR;
    hdma_sai_rx.Init.Priority = DMA_PRIORITY_HIGH;
    hdma_sai_rx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
    hdma_sai_rx.Init.FIFOThreshold = DMA_FIFO_THRESHOLD_FULL;
    hdma_sai_rx.Init.MemBurst = DMA_MBURST_SINGLE;
    hdma_sai_rx.Init.PeriphBurst = DMA_MBURST_SINGLE;
    hdma_sai_rx.Instance = AUDIO_IN_SAIx_DMAx_STREAM;

    /* Associate the DMA handle */
__HAL_LINKDMA(hsai, hdmarx, hdma_sai_rx);

/* Deinitialize the Stream for new transfer */
HAL_DMA_DeInit(&hdma_sai_rx);

/* Configure the DMA Stream */
HAL_DMA_Init(&hdma_sai_rx);

} }

/* SAI DMA IRQ Channel configuration */
HAL_NVIC_SetPriority(AUDIO_IN_SAIx_DMAx_IRQ,
AUDIO_IN_IRQ_PREPRIO, 0);
HAL_NVIC_EnableIRQ(AUDIO_IN_SAIx_DMAx_IRQ);

/* Audio INT IRQ Channel configuration */
HAL_NVIC_SetPriority(AUDIO_IN_INT_IRQ, AUDIO_IN_IRQ_PREPRIO, 0);
HAL_NVIC_EnableIRQ(AUDIO_IN_INT_IRQ);

/**
 * @brief De-Initializes SAI Audio IN MSP.
 * @param hsai: SAI handle
 * @param Params
 * @retval None
 */
static void SAI_AUDIO_IN_MspDeInit(SAI_HandleTypeDef *hsai,
void *Params)
{
    GPIO_InitTypeDef  gpio_init_structure;

    /* SAI DMA IRQ Channel deactivation */
    HAL_NVIC_DisableIRQ(AUDIO_IN_SAIx_DMAx_IRQ);

    // Other code...
}
if (hsai->Instance == AUDIO_IN_SAIx)
{
    /* Deinitialize the DMA stream */
    HAL_DMA_DeInit(hsai->hdmatx);
}

/* Disable SAI peripheral */
__HAL_SAI_DISABLE(hsai);

/* Deactivates CODEC_SAI pin SD by putting them in input mode */
gpio_init_structure.Pin = AUDIO_IN_SAIx_SD_PIN;
HAL_GPIO_DeInit(AUDIO_IN_SAIx_SD_GPIO_PORT, gpio_init_structure.Pin);

gpio_init_structure.Pin = AUDIO_IN_INT_GPIO_PIN;
HAL_GPIO_DeInit(AUDIO_IN_INT_GPIO_PORT, gpio_init_structure.Pin);

/* Disable SAI clock */
AUDIO_IN_SAIx_CLK_DISABLE();

/**
 * @brief Deinitializes SAI MSP.
 * @param hsai: SAI handle
 * @param Params
 * @retval None
 */
__weak void BSP_AUDIO_OUT_MspDeInit(SAI_HandleTypeDef *hsai,
                                   void *Params)
{
    GPIO_InitTypeDef  gpio_init_structure;

/* SAI DMA IRQ Channel deactivation */
HAL_NVIC_DisableIRQ(AUDIO_OUT_SAIx_DMAx_IRQ);

if(hsai->Instance == AUDIO_OUT_SAIx)
{
    /* Deinitialize the DMA stream */
    HAL_DMA_DeInit(hsai->hdmatx);
}

/* Disable SAI peripheral */
__HAL_SAI_DISABLE(hsai);

/* Deactivates CODEC_SAI pins FS, SCK, MCK and SD by putting them in input mode */
gpio_init_structure.Pin = AUDIO_OUT_SAIx_FS_PIN | AUDIO_OUT_SAIx_SCK_PIN | AUDIO_OUT_SAIx_SD_PIN;
HAL_GPIO_DeInit(AUDIO_OUT_SAIx_SD_FS_SCK_GPIO_PORT, gpio_init_structure.Pin);

gpio_init_structure.Pin = AUDIO_OUT_SAIx_MCLK_PIN;
HAL_GPIO_DeInit(AUDIO_OUT_SAIx_MCLK_GPIO_PORT, gpio_init_structure.Pin);

/* Disable SAI clock */
AUDIO_OUT_SAIx_CLK_DISABLE();

/* GPIO pins clock and DMA clock can be shut down in the applic
by surcharging this __weak function */

}
* @param sai: might be required to set a audio peripheral predivider if any.
* @param AudioFreq: Audio frequency used to play the audio stream.
* @param Params
* @note This API is called by BSP_AUDIO_OUT_Init() and BSP_AUDIO_OUT_SetFrequency()
* Being __weak it can be overwritten by the application
* @retval None
*/
__weak void BSP_AUDIO_OUT_ClockConfig(SAI_HandleTypeDef *hsai, uint32_t AudioFreq, void *Param)
{
    RCC_PermiphCLKInitTypeDef rcc_ex_clk_init_struct;
    HAL_RCCEx_GetPeriphCLKConfig(&rcc_ex_clk_init_struct);

    /* Set the PLL configuration according to the audio frequency */
    if((AudioFreq == AUDIO_FREQUENCY_11K) || (AudioFreq == AUDIO_FREQUENCY_22K) || (AudioFreq == AUDIO_FREQUENCY_44K))
    {
        /* Configure PLLSAI prescalers */
        /* PLLSAI_VCO: VCO_429M
        SAI_CLK(first level) = PLLSAI_VCO/PLLSAI Q = 429/2 = 214.5 Mhz
        SAI_CLK_x = SAI_CLK(first level)/PLLSAID IVQ = 214.5/19 = 11.289 Mhz */
        rcc_ex_clk_init_struct.PeriphClockSelection = RCC_PERIPHCLK_SAI1;
        rcc_ex_clk_init_struct.Sai1ClockSelection = RCC_SAI1CLKSOURCE_PLLI2S;
rcc_ex_clk_init_struct.PLLI2S.PLLI2SN = 429;
rcc_ex_clk_init_struct.PLLI2S.PLLI2SQ = 2;
rcc_ex_clk_init_struct.PLLI2SDivQ = 19;
HAL_RCCEx_PeriphCLKConfig(&rcc_ex_clk_init_struct);

else /* AUDIO_FREQUENCY_8K, AUDIO_FREQUENCY_16K, AUDIO_FREQUENCY_48K, AUDIO_FREQUENCY_96K */
{
    /* SAI clock config
     PLLSAI_VCO: VCO_344M
     SAI_CLK(first level) = PLLSAI_VCO/PLLSAIQ = 344/7 = 49.142 Mhz
     SAI_CLK_x = SAI_CLK(first level)/PLLSAIDIVQ = 49.142/1 = 49.142 Mhz */
    rcc_ex_clk_init_struct.PeriphClockSelection = RCC_PERIPHCLK_SAI1;
    rcc_ex_clk_init_struct.Sai1ClockSelection = RCC_SAI1CLKSOURCE_PLLI2S;
    rcc_ex_clk_init_struct.PLLI2S.PLLI2SN = 344;
    rcc_ex_clk_init_struct.PLLI2S.PLLI2SQ = 7;
    rcc_ex_clk_init_struct.PLLI2SDivQ = 1;
    HAL_RCCEx_PeriphCLKConfig(&rcc_ex_clk_init_struct);
}
/*****************************************************
/**
 * @brief Initializes the Audio Codec audio interface (SAI).
 * @param AudioFreq: Audio frequency to be configured for the SAI peripheral.
 * @note The default SlotActive configuration is set to CODEC_AUDIOFRAME_SLOT_0123
 * and user can update this configuration using
 * @retval None
 */

static void SAIx_Out_Init(uint32_t AudioFreq)
{
    /* Initialize the haudio_out_sai Instance parameter */
    haudio_out_sai.Instance = AUDIO_OUT_SAIx;

    /* Disable SAI peripheral to allow access to SAI internal registers */
    __HAL_SAI_DISABLE(&haudio_out_sai);

    /* Configure SAI_Block_x LSBFirst: Disabled DataSize: 16 */
    haudio_out_sai.Init.MonoStereoMode = SAI_STEREOMODE;
    haudio_out_sai.Init.AudioFrequency = AudioFreq;
    haudio_out_sai.Init.AudioMode = SAI_MODEMASTER_TX;
    haudio_out_sai.Init.NoDivider = SAI_MASTER_DIVIDER_ENABLED;
haudio_out_sai.Init.Protocol = SAI_FREE_PRTOCOL;
haudio_out_sai.Init.DataSize = SAI_DATASIZE_16;
haudio_out_sai.Init.FirstBit = SAI_FIRSTBIT_MSB;
haudio_out_sai.Init.ClockStrobing = SAI_CLOCKSTROBING_RISINGEDGE;
haudio_out_sai.Init.Synchro = SAIASYNCHRONOUS;
haudio_out_sai.Init.OutputDrive = SAI_OUTPUTDRIVE_ENABLED;
haudio_out_sai.Init.FIFOThreshold = SAI_FIFO_THRESHOLD_1QF;
haudio_out_sai.Init.SynchroExt = SAI_SYNCEXT_DISABLE;
haudio_out_sai.Init.CompandingMode = SAI_NOCOMPANDING;
haudio_out_sai.Init.TriState = SAI_OUTPUTNOTRELEASED;
haudio_out_sai.Init.Mckdiv = 0;

/* Configure SAI_Block_x Frame */
Frame Length: 64
Frame active Length: 32
FS Definition: Start frame + Channel Side identification
FS Polarity: FS active Low
FS Offset: FS asserted one bit before the first bit of slot 0 */
haudio_out_sai.FrameInit.FrameLength = 128;
haudio_out_sai.FrameInit.ActiveFrameLength = 64;
haudio_out_sai.FrameInit.FSDefinition = SAI_FS_CHANNEL_IDENTIFICATION;
haudio_out_sai.FrameInit.FSPolarity = SAI_
FS_ACTIVE_LOW;
00937  haudio_out_sai.FrameInit.FSOffset = SAI_FS_BEFORFIRSTBIT;
00938
00939  /* Configure SAI Block_x Slot */
00940  Slot First Bit Offset: 0
00941  Slot Size : 16
00942  Slot Number: 4
00943  Slot Active: All slot actives */
00944  haudio_out_sai.SlotInit.FirstBitOffset = 0;
00945  haudio_out_sai.SlotInit.SlotSize = SAI_SLOTSIZE_DATASIZE;
00946  haudio_out_sai.SlotInit.SlotNumber = 4;
00947  haudio_out_sai.SlotInit.SlotActive = CODEC_AUDIOFRAME_SLOT_0123;
00948
00949  HAL_SAI_Init(&haudio_out_sai);
00950
00951  /* Enable SAI peripheral to generate MCLK */
00952  __HAL_SAI_ENABLE(&haudio_out_sai);
00953  }
00954
00955  /**
00956  * @brief Deinitializes the Audio Codec audio interface (SAI).
00957  * @retval None
00958  */
00959  static void SAIx_Out_DeInit(void)
00960  {
00961  /* Initialize the haudio_out_sai Instance parameter */
00962  haudio_out_sai.Instance = AUDIO_OUT_SAIx;
00963
00964  /* Disable SAI peripheral */
00965  __HAL_SAI_DISABLE(&haudio_out_sai);
HAL_SAI_DeInit(&haudio_out_sai);

/**
 * @brief Initializes the Audio Codec audio interface (SAI).
 * @param AudioFreq: Audio frequency to be configured for the SAI peripheral.
 * @note The default SlotActive configuration is set to CODEC_AUDIOFRAME_SLOT_0123 and user can update this configuration using
 * @retval None
 */
static void SAIx_In_Init(uint32_t AudioFreq)
{
    /* Initialize SAI1 block A in MASTER TX */
    /* Initialize the haudio_out_sai Instance parameter */
    haudio_out_sai.Instance = AUDIO_OUT_SAIx;

    /* Disable SAI peripheral to allow access to SAI internal registers */
    __HAL_SAI_DISABLE(&haudio_out_sai);

    /* Configure SAI_Block_x LSBFirst: Disabled DataSize: 16 */
    haudio_out_sai.Init.MonoStereoMode = SAI_STEREOMODE;
    haudio_out_sai.Init.AudioFrequency = AudioFreq;
    haudio_out_sai.Init.AudioMode = SAI_MODEMASTER_RX;
    haudio_out_sai.Init.NoDivider = SAI_MASTERDIVIDER_ENABLE;
haudio_out_sai.Init.Protocol = SAI_FREE_PROTOCOL;
haudio_out_sai.Init.DataSize = SAI_DATASIZE_16;
haudio_out_sai.Init.FirstBit = SAIIRSTBIT_MSB;
haudio_out_sai.Init.ClockStrobing = SAI_LOCKSTROBING_FALLINGEDGE;
haudio_out_sai.Init.Synchro = SAI_SYNCHRONOUS;
haudio_out_sai.Init.OutputDrive = SAI_OUTPUTDRIVE_ENABLE;
haudio_out_sai.Init.FIFOThreshold = SAI_FIFO_THRESHOLD_1QF;
haudio_out_sai.Init.SynchroExt = SAI_SYNCEXT_DISABLE;
haudio_out_sai.Init.CompressingMode = SAI_NOCOMPANDING;
haudio_out_sai.Init.TriState = SAI_OUTPUT_NOTRELEASED;
haudio_out_sai.Init.Mckdiv = 0;

/* Configure SAI_Block_x Frame */
Frame Length: 64
Frame active Length: 32
FS Definition: Start frame + Channel Side identification
FS Polarity: FS active Low
FS Offset: FS asserted one bit before the first bit of slot 0 */
haudio_out_sai.FrameInit.FrameLength = 64;
haudio_out_sai.FrameInit.ActiveFrameLength = 32;
haudio_out_sai.FrameInit.FSDefinition = SAI_FS_CHANNEL_IDENTIFICATION;
haudio_out_sai.FrameInit.FSPolarity
= SAI_FS_ACTIVE_LOW;
01015  haudio_out_sai.FrameInit.FSOffset = SAI_FS_BEFOREFIRSTBIT;
01016
01017  /* Configure SAI Block_x Slot */
01018  Slot First Bit Offset: 0
01019  Slot Size : 16
01020  Slot Number: 4
01021  Slot Active: All slot actives */
01022  haudio_out_sai.SlotInit.FirstBitOffset = 0;
01023  haudio_out_sai.SlotInit.SlotSize = SAI_SLOTSIZE_DATASIZE;
01024  haudio_out_sai.SlotInit.SlotNumber = 4;
01025  haudio_out_sai.SlotInit.SlotActive = CODEC_AUDIOFRAME_SLOT_02;
01026  HAL_SAI_Init(&haudio_out_sai);
01027
01028  /* Initialize SAI1 block B in SLAVE RX synchronous from SAI1 block A */
01029  /* Initialize the haudio_in_sai Instance parameter */
01030  haudio_in_sai.Instance = AUDIO_IN_SAIx;
01031  haudio_in_sai.Init.MonoStereoMode = SAI_STEREOMODE;
01032  __HAL_SAI_DISABLE(&haudio_in_sai);
01033  /* Disable SAI peripheral to allow access to SAI internal registers */
01034  haudio_in_sai.Init.AudioFrequency = AudioFreq;
01041  haudio_in_sai.Init.AudioMode = SAI_MO
DESLAVE_RX;
01042  haudio_in_sai.Init.NoDivider = SAI_MA
STERDIVIDER_ENABLE;
01043  haudio_in_sai.Init.Protocol = SAI_FR
EE_PROTOCOL;
01044  haudio_in_sai.Init.DataSize = SAI_DA
TASIZE_16;
01045  haudio_in_sai.Init.FirstBit = SAI_FI
RSTBIT_MSB;
01046  haudio_in_sai.Init.ClockStrobing = SAI_CL
OCKSTROBING_RISINGEDGE;
01047  haudio_in_sai.Init.Synchro = SAI_SY
NCHRONOUS;
01048  haudio_in_sai.Init.OutputDrive = SAI_OU
PUTDRIVE_DISABLE;
01049  haudio_in_sai.Init.FIFOThreshold = SAI_FI
FO_THRESHOLD_1QF;
01050  haudio_in_sai.Init.SynchroExt = SAI_SY
EXT_DISABLE;
01051  haudio_in_sai.Init.CompandingMode = SAI_NO
COMPANDING;
01052  haudio_in_sai.Init.TriState = SAI_OU
PUT_RELEASED;
01053  haudio_in_sai.Init.Mckdiv = 0;
01054
01055  /* Configure SAI_Block_x Frame
01056  Frame Length: 64
01057  Frame active Length: 32
01058  FS Definition: Start frame + Channel Side
    identification
01059  FS Polarity: FS active Low
01060  FS Offset: FS asserted one bit before the
    first bit of slot 0 */
01061  haudio_in_sai.FrameInit.FrameLength
 = 64;
01062  haudio_in_sai.FrameInit.ActiveFrameLength
```c
01062 = 32;
01063    haudio_in_sai.FrameInit.FSDefinition = SAI_FS_CHANNEL_IDENTIFICATION;
01064    haudio_in_sai.FrameInit.FSPolarity = SAI_FS_ACTIVE_LOW;
01065    haudio_in_sai.FrameInit.FSOffset = SAI_FS_BEFORFIRSTBIT;
01066
01067    /* Configure SAI Block_x Slot
01068    Slot First Bit Offset: 0
01069    Slot Size : 16
01070    Slot Number: 4
01071    Slot Active: All slot active */
01072    haudio_in_sai.SlotInit.FirstBitOffset = 0;
01073    haudio_in_sai.SlotInit.SlotSize = SAI_SLOTSIZE_DATASIZE;
01074    haudio_in_sai.SlotInit.SlotNumber = 4;
01075    haudio_in_sai.SlotInit.SlotActive = CO DEC_AUDIOFRAME_SLOT_02;
01076
01077    HAL_SAI_Init(&haudio_in_sai);
01078
01079    /* Enable SAI peripheral */
01080    __HAL_SAI_ENABLE(&haudio_in_sai);
01081
01082    /* Enable SAI peripheral to generate MCLK */
01083    __HAL_SAI_ENABLE(&haudio_out_sai);
01084 }
01085
01086 /**
01087    * @brief Deinitializes the output Audio Codec audio interface (SAI).
01088    * @retval None
01089    */
01090    static void SAIx_In_DeInit(void)
01091 {```
/* Initialize the haudio_in_sai Instance parameter */
haudio_in_sai.Instance = AUDIO_IN_SAIx;
haudio_out_sai.Instance = AUDIO_OUT_SAIx;
/* Disable SAI peripheral */
__HAL_SAI_DISABLE(&haudio_in_sai);
HAL_SAI_DeInit(&haudio_in_sai);
HAL_SAI_DeInit(&haudio_out_sai);
}
/** @defgroup STM32F769I_DISCOVERY_AUDIO_In_Private_Functions STM32F769I_DISCOVERY_AUDIO_In_Private Functions */
/** @brief Initialize wave recording. */
/** @param AudioFreq: Audio frequency to be configured for the DFSDM peripheral. */
/** @param BitRes: Audio frequency to be configured for the DFSDM peripheral. */
/** @param ChnlNbr: Audio frequency to be configured for the DFSDM peripheral. */
/** @retval AUDIO_OK if correct communication, else wrong communication */

uint8_t BSP_AUDIO_IN_Init(uint32_t AudioFreq, uint32_t BitRes, uint32_t ChnlNbr)
{
    return BSP_AUDIO_IN_InitEx(INPUT_DEVICE_DIGITAL_MIC, AudioFreq, BitRes, ChnlNbr);
/**
* @brief Initialize wave recording.
* @param InputDevice: INPUT_DEVICE_DIGITAL_MIC or INPUT_DEVICE_ANALOG_MIC.
* @param AudioFreq: Audio frequency to be configured.
* @param BitRes: Audio bit resolution to be configured.
* @param ChnlNbr: Number of channel to be configured.
* @retval AUDIO_OK if correct communication, else wrong communication
*/

uint8_t BSP_AUDIO_IN_InitEx(uint16_t InputDevice, uint32_t AudioFreq, uint32_t BitRes, uint32_t ChnlNbr)
{
    uint8_t ret = AUDIO_ERROR;
    AudioIn_Device = InputDevice;

    if (InputDevice == INPUT_DEVICE_DIGITAL_MIC)
    {
        AudioIn_ChannelNumber = ChnlNbr;
        /* PLL clock is set depending by the AudioFreq (44.1khz vs 48khz groups) */
        BSP_AUDIO_IN_ClockConfig(&hAudioInTopLeftFilter, AudioFreq, NULL);
        /* Init the SAI MSP: this __weak function can be redefined by the application*/
        BSP_AUDIO_IN_MspInit();

        /* Initializes DFSDM peripheral */
        DFSDMx_Init(AudioFreq);
    ret = AUDIO_OK;
    }
    else
    {
        /* Disable SAI */
        SAIx_In_DeInit();
        /* PLL clock is set depending by the AudioFreq (44.1khz vs 48khz groups) */
        BSP_AUDIO_OUT_ClockConfig(&haudio_in_sai, AudioFreq, NULL);
        haudio_in_sai.Instance = AUDIO_IN_SAIx;
        if(HAL_SAI_GetState(&haudio_in_sai) == HAL_SAI_STATE_RESET)
        {
            BSP_AUDIO_OUT_MspInit(&haudio_in_sai, NULL);
            BSP_AUDIO_IN_MspInit();
        }
        SAIx_In_Init(AudioFreq);
        if((wm8994_drv.ReadID(AUDIO_I2C_ADDRESS) == WM8994_ID)
        {
            /* Reset the Codec Registers */
            wm8994_drv.Reset(AUDIO_I2C_ADDRESS);
            /* Initialize the audio driver structure */
            audio_drv = &wm8994_drv;
            ret = AUDIO_OK;
        }
        else
        {
            ret = AUDIO_ERROR;
        }
    }
if (ret == AUDIO_OK)
{
    /* Initialize the codec internal registers */
audio_drv->Init(AUDIO_I2C_ADDRESS, InputDevice, 100, AudioFreq);
}

/* Return AUDIO_OK when all operations are correctly done */
return ret;

/**
 * @brief Allocate channel buffer scratch
 * @param pScratch : pointer to scratch tables.
 * @param size of scratch buffer
 */
uint8_t BSP_AUDIO_IN_AllocScratch (int32_t *pScratch, uint32_t size)
{
    uint32_t idx;
    ScratchSize = (size / AudioIn_ChannelNumber);

    /* copy scratch pointers */
    for (idx = 0; idx < AudioIn_ChannelNumber; idx++)
    {
        pScratchBuff[idx] = (int32_t *)(pScratch + (idx * ScratchSize));
    }
    /* Return AUDIO_OK */
return AUDIO_OK;
}

/**
 * @brief Return audio in channel number
 * @retval Number of channel
 */
uint8_t BSP_AUDIO_IN_GetChannelNumber(void)
{
    return AudioIn_ChannelNumber;
}

/**
 * @brief Start audio recording.
 * @param pbuf: Main buffer pointer for the recorded data storing
 * @param size: Current size of the recorded buffer
 * @retval AUDIO_OK if correct communication, else wrong communication
 */
uint8_t BSP_AUDIO_IN_Record(uint16_t* pbuf, uint32_t size)
{
    if (AudioIn_Device == INPUT_DEVICE_DIGITAL_MIC)
    {
        hAudioIn.pRecBuf = pbuf;
        hAudioIn.RecSize = size;
        /* Reset Application Buffer Trigger */
        AppBuffTrigger = 0;
        AppBuffHalf = 0;
        if (AudioIn_ChannelNumber > 2)
        {
            /* Call the Media layer start function for bottom right channel */
        }
if(HAL_OK != HAL_DFSDM_FilterRegularStart_DMA(&hAudioInButtomRightFilter, pScratchBuff[2], ScratchSize))
{
    return AUDIO_ERROR;
}

/* Call the Media layer start function for buttom left channel */
if(HAL_OK != HAL_DFSDM_FilterRegularStart_DMA(&hAudioInButtomLeftFilter, pScratchBuff[3], ScratchSize))
{
    return AUDIO_ERROR;
}

/* Call the Media layer start function for top right channel */
if(HAL_OK != HAL_DFSDM_FilterRegularStart_DMA(&hAudioInTopRightFilter, pScratchBuff[0], ScratchSize))
{
    return AUDIO_ERROR;
}

/* Call the Media layer start function for top left channel */
if(HAL_OK != HAL_DFSDM_FilterRegularStart_DMA(&hAudioInTopLeftFilter, pScratchBuff[1], ScratchSize))
{
    return AUDIO_ERROR;
}

else
{

/* Start the process receive DMA */
if(HAL_OK != HAL_SAI_Receive_DMA(&haudio_in_sai, (uint8_t*)pbuf, size))
{
    return AUDIO_ERROR;
}

/* Return AUDIO_OK when all operations are correctly done */
return AUDIO_OK;

/**
 * @brief Stop audio recording.
 * @retval AUDIO_OK if correct communication, else wrong communication
 */
uint8_t BSP_AUDIO_IN_Stop(void) {
    if (AudioIn_Device == INPUT_DEVICE_DIGITAL_MIC)
    {
        AppBuffTrigger = 0;
        AppBuffHalf = 0;
        if(AudioIn_ChannelNumber > 2)
        {
            /* Call the Media layer stop function for buttom right channel */
            if(HAL_OK != HAL_DFSDM_FilterRegularStop_DMA(&hAudioInButtomRightFilter))
            {
                return AUDIO_ERROR;
            }
        }
    }
    /* Call the Media layer stop function for buttom left channel */
if(HAL_OK != HAL_DFSDM_FilterRegularStop_DMA(&hAudioInButtomLeftFilter))
{
    return AUDIO_ERROR;
}

/* Call the Media layer stop function for top right channel */
if(HAL_OK != HAL_DFSDM_FilterRegularStop_DMA(&hAudioInTopRightFilter))
{
    return AUDIO_ERROR;

/* Call the Media layer stop function for top left channel */
if(HAL_OK != HAL_DFSDM_FilterRegularStop_DMA(&hAudioInTopLeftFilter))
{
    return AUDIO_ERROR;

else
{
    /* Call the Media layer stop function */
    HAL_SAI_DMAStop(&haudio_in_sai);

    /* Call Audio Codec Stop function */
    if(audio_drv->Stop(AUDIO_I2C_ADDRESS, CODEC_PDWN_HW) != 0)
    {
        return AUDIO_ERROR;
    }
    else
    {
        /* Wait at least 100us */
        }
HAL_Delay(1);

/* Return AUDIO_OK when all operations are correctly done */
return AUDIO_OK;
}

/* Return AUDIO_OK when all operations are correctly done */
return AUDIO_OK;

/**
 * @brief  Pause the audio file stream.
 * @retval AUDIO_OK if correct communication, else wrong communication
 */
uint8_t BSP_AUDIO_IN_Pause(void)
{
  if(AudioIn_ChannelNumber > 2)
  {
    /* Call the Media layer stop function */
    if(HAL_OK != HAL_DFSDM_FilterRegularStop_DMA(&hAudioInButtomRightFilter))
      {
        return AUDIO_ERROR;
      }
    /* Call the Media layer stop function */
    if(HAL_OK != HAL_DFSDM_FilterRegularStop_DMA(&hAudioInButtomLeftFilter))
      {
        return AUDIO_ERROR;
      }
  }
  /* Call the Media layer stop function */
  if(HAL_OK != HAL_DFSDM_FilterRegularStop_D
MA(&hAudioInTopRightFilter))
01356    {  
01357        return AUDIO_ERROR;
01358    }  
01359  
01360  /* Call the Media layer stop function */
01361  if(HAL_OK != HAL_DFSDM_FilterRegularStop_DMA(&hAudioInTopLeftFilter))
01362    {  
01363        return AUDIO_ERROR;
01364    }  
01365  
01366  /* Return AUDIO_OK when all operations are correctly done */
01367  return AUDIO_OK;
01368  }
01369
01370  /**
01371  * @brief Resume the audio file stream.
01372  * @retval AUDIO_OK if correct communication, else wrong communication
01373  */
01374  uint8_t BSP_AUDIO_IN_Resume(void)
01375    {  
01376        if(AudioIn_ChannelNumber > 2)
01377        {  
01378            /* Call the Media layer start function for bottom right channel */
01379            if(HAL_OK != HAL_DFSDM_FilterRegularStart_DMA(&hAudioInButtomRightFilter, pScratchBuff[2], ScratchSize))
01380                {  
01381                    return AUDIO_ERROR;
01382                }  
01383        }  
01384        /* Call the Media layer start function for bottom left channel */
if (HAL_OK != HAL_DFSDM_FilterRegularStart_DMA(&hAudioInButtomLeftFilter, pScratchBuff[3], ScratchSize))
    return AUDIO_ERROR;
}
/* Call the Media layer start function for top right channel */
if (HAL_OK != HAL_DFSDM_FilterRegularStart_DMA(&hAudioInTopRightFilter, pScratchBuff[0], ScratchSize))
    return AUDIO_ERROR;
}
/* Call the Media layer start function for top left channel */
if (HAL_OK != HAL_DFSDM_FilterRegularStart_DMA(&hAudioInTopLeftFilter, pScratchBuff[1], ScratchSize))
    return AUDIO_ERROR;
}
/* Return AUDIO_OK when all operations are correctly done */
return AUDIO_OK;
}
/**
 * @brief Deinit the audio IN peripherals.
 * @retval None
 */
void BSP_AUDIO_IN_DeInit(void)
{
    BSP_AUDIO_IN_MspDeInit();
if (AudioIn_Device == INPUT_DEVICE_DIGITAL_MIC)
{
    DFSDMx_DeInit();
}
else
{
    SAIx_In_DeInit();
}

/**
 * @brief	Regular conversion complete call back.
 * @note	In interrupt mode, user has to read conversion value in this function
 * using HAL_DFSDM_FilterGetRegular Value.
 * @param hdfsdm_filter : DFSDM filter handle.
 * @retval None
 */

void HAL_DFSDM_FilterRegConvCpltCallback(DFSDM_Filter_HandleTypeDef hdfsdm_filter)
{
    uint32_t index = 0;
    if(hdfsdm_filter == &hAudioInTopLeftFilter)
    {
        DmaTopLeftRecCplt = 1;
    }
    else if(hdfsdm_filter == &hAudioInTopRightFilter)
    {
        DmaTopRightRecCplt = 1;
    }
else if(hdfsdm_filter == &hAudioInBottomLeftFilter)
{
    DmaBottomLeftRecCplt = 1;
}
else
{
    DmaBottomRightRecCplt = 1;
}
}

if(AudioIn_ChannelNumber > 2)
{
    if((DmaTopLeftRecCplt == 1) && (DmaTopRightRecCplt == 1) && (DmaBottomLeftRecCplt == 1) && (DmaBottomRightRecCplt == 1))
    {
        for(index = (ScratchSize/2); index < ScratchSize; index++)
        {
            hAudioIn.pRecBuf[AppBuffTrigger] = (uint16_t)(SaturaLH((pScratchBuff[1][index] >> 8), -32760, 32760));
            hAudioIn.pRecBuf[AppBuffTrigger + 1] = (uint16_t)(SaturaLH((pScratchBuff[0][index] >> 8), -32760, 32760));
            hAudioIn.pRecBuf[AppBuffTrigger + 2] = (uint16_t)(SaturaLH((pScratchBuff[3][index] >> 8), -32760, 32760));
            hAudioIn.pRecBuf[AppBuffTrigger + 3] = (uint16_t)(SaturaLH((pScratchBuff[2][index] >> 8), -32760, 32760));
            AppBuffTrigger +=4;
        }
        DmaTopLeftRecCplt = 0;
        DmaTopRightRecCplt = 0;
        DmaBottomLeftRecCplt = 0;
DmaButtomRightRecCplt = 0;

if((DmaTopLeftRecCplt == 1) && (DmaTopRightRecCplt == 1))
{
    for(index = (ScratchSize/2) ; index < ScratchSize; index++)
    {
        hAudioIn.pRecBuf[AppBuffTrigger] = (uint16_t)(SaturaLH((pScratchBuff[1][index] >> 8), -32760, 32760));
        hAudioIn.pRecBuf[AppBuffTrigger + 1] = (uint16_t)(SaturaLH((pScratchBuff[0][index] >> 8), -32760, 32760));
        AppBuffTrigger += 2;
    }
    DmaTopLeftRecCplt = 0;
    DmaTopRightRecCplt = 0;
}

/* Call Half Transfer Complete callback */
if((AppBuffTrigger == hAudioIn.RecSize/2) && (AppBuffHalf == 0))
{
    AppBuffHalf = 1;
    BSP_AUDIO_IN_HalfTransfer_CallBack();
}

/* Call Transfer Complete callback */
if(AppBuffTrigger == hAudioIn.RecSize)
{
    /* Reset Application Buffer Trigger */
    AppBuffTrigger = 0;
    AppBuffHalf = 0;
/* Call the record update function to get the next buffer to fill and its size (size is ignored) */
BSP_AUDIO_IN_TransferComplete_CallBack();
}

/**
 * @brief Half regular conversion complete callback.
 * @param hdfsdm_filter : DFSDM filter handle.
 * @retval None
 */
void HAL_DFSDM_FilterRegConvHalfCpltCallback(DFSDM_Filter_HandleTypeDef *hdfsdm_filter)
{
    uint32_t index = 0;

    if (hdfsdm_filter == &hAudioInTopLeftFilter)
    {
        DmaTopLeftRecHalfCplt = 1;
    }
    else if (hdfsdm_filter == &hAudioInTopRightFilter)
    {
        DmaTopRightRecHalfCplt = 1;
    }
    else if (hdfsdm_filter == &hAudioInButtomLeftFilter)
    {
        DmaButtomLeftRecHalfCplt = 1;
    }
    else
    {

DmaButtomRightRecHalfCplt = 1;
}

if (AudioIn_ChannelNumber > 2) {
  if((DmaTopLeftRecHalfCplt == 1) && (DmaTopRightRecHalfCplt == 1) && (DmaBottomLeftRecHalfCplt == 1) && (DmaBottomRightRecHalfCplt == 1))
  {
    for(index = 0 ; index < ScratchSize/2; index++)
    {
      hAudioIn.pRecBuf[AppBuffTrigger] = (uint16_t)(SaturaLH((pScratchBuff[1][index] >> 8), -32760, 32760));
      hAudioIn.pRecBuf[AppBuffTrigger + 1] = (uint16_t)(SaturaLH((pScratchBuff[0][index] >> 8), -32760, 32760));
      hAudioIn.pRecBuf[AppBuffTrigger + 2] = (uint16_t)(SaturaLH((pScratchBuff[3][index] >> 8), -32760, 32760));
      hAudioIn.pRecBuf[AppBuffTrigger + 3] = (uint16_t)(SaturaLH((pScratchBuff[2][index] >> 8), -32760, 32760));
      AppBuffTrigger +=4;
    }
    DmaTopLeftRecHalfCplt = 0;
    DmaTopRightRecHalfCplt = 0;
    DmaBottomLeftRecHalfCplt = 0;
    DmaBottomRightRecHalfCplt = 0;
  }
else
  {
    if((DmaTopLeftRecHalfCplt == 1) && (DmaTopRightRecHalfCplt == 1))
    {
      // Code...
    }
  }
}
for (index = 0 ; index < ScratchSize/2; index++)
{
    hAudioIn.pRecBuf[AppBuffTrigger] = (uint16_t)(SaturaLH((pScratchBuff[1][index] >> 8), -32760, 32760));
    hAudioIn.pRecBuf[AppBuffTrigger + 1] = (uint16_t)(SaturaLH((pScratchBuff[0][index] >> 8), -32760, 32760));
    AppBuffTrigger += 2;
}

DmaTopLeftRecHalfCplt = 0;
DmaTopRightRecHalfCplt = 0;
}

/* Call Half Transfer Complete callback */
if((AppBuffTrigger == hAudioIn.RecSize/2) && (AppBuffHalf == 0))
{
    AppBuffHalf = 1;
    BSP_AUDIO_IN_HalfTransfer_CallBack();
}

/* Call Transfer Complete callback */
if(AppBuffTrigger == hAudioIn.RecSize)
{
    /* Reset Application Buffer Trigger */
    AppBuffTrigger = 0;
    AppBuffHalf = 0;
    /* Call the record update function to get the next buffer to fill and its size (size is ignored) */
    BSP_AUDIO_IN_TransferComplete_CallBack();
}

/**
 * @brief Half reception complete callback.
 *
 * @param hsai : SAI handle.
 *
 * @retval None
 */

void HAL_SAI_RxHalfCpltCallback(SAI_HandleTypeDef *hsai)
{
    /* Manage the remaining file size and new address offset: This function should be coded by user (its prototype is already declared in stm32769i_discovery_audio.h) */
    BSP_AUDIO_IN_HalfTransfer_CallBack();
}

/**
 * @brief Reception complete callback.
 *
 * @param hsai : SAI handle.
 *
 * @retval None
 */

void HAL_SAI_RxCpltCallback(SAI_HandleTypeDef *hsai)
{
    /* Call the record update function to get the next buffer to fill and its size (size is ignored) */
    BSP_AUDIO_IN_TransferComplete_CallBack();
}

/**
 * @brief User callback when record buffer is filled.
 *
 * @retval None
 */

__weak void BSP_AUDIO_IN_TransferComplete_CallBack()
void llBack()
01606  {
01607    /* This function should be implemented by
the user application.
01608    It is called into this driver when the
current buffer is filled
01609    to prepare the next buffer pointer and
its size. */
01610  }
01611
01612  /**
01613    * @brief Manages the DMA Half Transfer complete event.
01614    * @retval None
01615    */
01616    __weak void BSP_AUDIO_IN_HalfTransfer_CallBack(void)
01617  {
01618    /* This function should be implemented by
the user application.
01619    It is called into this driver when the
current buffer is filled
01620    to prepare the next buffer pointer and
its size. */
01621  }
01622
01623  /**
01624    * @brief Audio IN Error callback function.
01625    * @retval None
01626    */
01627    __weak void BSP_AUDIO_IN_Error_CallBack(void)
01628  {
01629    /* This function is called when an Interrupt
due to transfer error on or peripheral
01630    error occurs. */


/**
 * @brief Initialize BSP_AUDIO_IN MSP.
 * @retval None
 */
__weak void BSP_AUDIO_IN_MspInit(void)
{
    if (AudioIn_Device == INPUT_DEVICE_DIGITAL_MIC)
    {
        /* MSP channels initialization */
        DFSDMx_ChannelMspInit();
        /* MSP filters initialization */
        DFSDMx_FilterMspInit();
    }
    else
    {
        SAI_AUDIO_IN_MspInit(&haudio_in_sai, NULL);
    }
}

/**
 * @brief DeInitialize BSP_AUDIO_IN MSP.
 * @retval None
 */
__weak void BSP_AUDIO_IN_MspDeInit(void)
{
    if (AudioIn_Device == INPUT_DEVICE_DIGITAL_MIC)
    {
        /* MSP channels initialization */
        DFSDMx_ChannelMspDeInit();
        /* MSP filters initialization */
        DFSDMx_FilterMspDeInit();
    }
}
else {
    SAI_AUDIO_IN_MspDeInit(&haudio_in_sai, NULL);
}

/**
 * @brief Clock Config.
 * @param hdfsdm_filter: might be required to set audio peripheral predivider if any.
 * @param AudioFreq: Audio frequency used to play the audio stream.
 * @param Params
 * @note This API is called by BSP_AUDIO_IN_Init()
 *       Being __weak it can be overwritten by the application
 * @retval None
 */
__weak void BSP_AUDIO_IN_ClockConfig(DFSDM_Filter_HandleTypeDef *hdfsdm_filter, uint32_t AudioFreq, void *Params) {
    RCC_PeriphCLKInitTypeDef rcc_ex_clk_init_struct;

    HAL_RCCEx_GetPeriphCLKConfig(&rcc_ex_clk_init_struct);

    /* Set the PLL configuration according to the audio frequency */
    if((AudioFreq == AUDIO_FREQUENCY_11K) || (AudioFreq == AUDIO_FREQUENCY_22K) || (AudioFreq == AUDIO_FREQUENCY_44K)) {
        /* Configure PLLSAI prescalers */
/* PLLI2S_VCO: VCO_429M */
SAI_CLK(first level) = PLLI2S_VCO/PLLI2SQ
Q = 429/2 = 214.5 Mhz
SAI_CLK_x = SAI_CLK(first level)/PLLI2SDIVQ = 214.5/19 = 11.289 Mhz */

else /* AUDIO_FREQUENCY_8K, AUDIO_FREQUENCY_16K, AUDIO_FREQUENCY_32K, AUDIO_FREQUENCY_48K, AUDIO_FREQUENCY_96K */
{
    /* SAI clock config */
    PLLI2S_VCO: VCO_344M
    SAI_CLK(first level) = PLLI2S_VCO/PLLI2SQ
Q = 344/7 = 49.142 Mhz
SAI_CLK_x = SAI_CLK(first level)/PLLI2SDIVQ = 49.142/1 = 49.142 Mhz */

    rcc_ex_clk_init_struct.PeriphClockSelection = RCC_PERIPHCLK_SAI2;
    rcc_ex_clk_init_struct.Sai2ClockSelection = RCC_SAI2CLKSOURCE_PLLI2S;
    rcc_ex_clk_init_struct.PLLI2S.PLLI2SN = 344;
    rcc_ex_clk_init_struct.PLLI2S.PLLI2SQ = 7;
    HAL_RCCEx_PeriphCLKConfig(&rcc_ex_clk_init_struct);
}
rcc_ex_clk_init_struct.PLLI2SDivQ = 1;
HAL_RCCEx_PeriphCLKConfig(&rcc_ex_clk_init_struct);
}

rcc_ex_clk_init_struct.PeriphClockSelection = RCC_PERIPHCLK_DFSDM1_AUDIO;
rcc_ex_clk_init_struct.Dfsdm1AudioClockSelection = RCC_DFSDM1AUDIOCLKSOURCE_SAI2;
HAL_RCCEx_PeriphCLKConfig(&rcc_ex_clk_init_struct);
}

/****************************
* Static Functions
****************************
static uint8_t DFSDMx_Init(uint32_t AudioFreq)

01734 { 
01735 /***************************************************************************/
01736 /***************************************************************************/
01737 /******************************************************************************/
01738 /* CHANNEL 1 configuration */
01739 __HAL_DFSDM_CHANNEL_RESET_HANDLE_STATE(&hAudioInTopLeftChannel);
01740 hAudioInTopLeftChannel.Instance = DFSDM1_Channel1;
01741 hAudioInTopLeftChannel.Init.OutputClock.Activation = ENABLE;
01742 hAudioInTopLeftChannel.Init.OutputClock.Selection = DFSDM_CHANNEL_OUTPUT_CLOCK_AUDIO;
01743 /* Set the DFSDM clock OUT audio frequency configuration */
01744 hAudioInTopLeftChannel.Init.OutputClock.Divider = DFSDM_CLOCK_DIVIDER(AudioFreq);
01745 hAudioInTopLeftChannel.Init.Input.Multiplex = DFSDM_CHANNEL_EXTERNAL_INPUTS;
01746 hAudioInTopLeftChannel.Init.Input.DataPacking = DFSDM_CHANNEL_STANDARD_MODE;
01747 hAudioInTopLeftChannel.Init.Input.Pins = DFSDM_CHANNEL_SAME_CHANNEL_PINS;
01748 /* Request to sample stable data for LEFT micro on Rising edge */
01749 hAudioInTopLeftChannel.Init.SerialInterface.Type = DFSDM_CHANNEL_SPI_RISING;
01750 hAudioInTopLeftChannel.Init.SerialInterface.SpiClock = DFSDM_CHANNEL_SPI_CLOCK_INTERNAL;
01751 hAudioInTopLeftChannel.Init.Awd.FilterOrder = DFSDM_CHANNEL_FASTSINC_ORDER;
01752 hAudioInTopLeftChannel.Init.Awd.Oversampling = 10;
01753 hAudioInTopLeftChannel.Init.Offset
hAudioInTopLeftChannel.Init.RightBitShift = DFSDM_RIGHT_BIT_SHIFT(AudioFreq);
if (HAL_OK != HAL_DFSDM_ChannelInit(&hAudioInTopLeftChannel)) {
    return AUDIO_ERROR;
}

/* CHANNEL 0 configuration */
__HAL_DFSDM_CHANNEL_RESET_HANDLE_STATE(&hAudioInTopRightChannel);
hAudioInTopRightChannel.Instance = DFSDM1_Channel0;
hAudioInTopRightChannel.Init.OutputClock.Activation = ENABLE;
hAudioInTopRightChannel.Init.OutputClock.Selection = DFSDM_CHANNEL_OUTPUT_CLOCK_AUDIO;
/* Set the DFSDM clock OUT audio frequency configuration */
hAudioInTopRightChannel.Init.OutputClock.Divider = DFSDM_CLOCK_DIVIDER(AudioFreq);
hAudioInTopRightChannel.Init.Input.Multiplexer = DFSDM_CHANNELTERNAL_INPUTS;
hAudioInTopRightChannel.Init.Input.DataPacking = DFSDM_CHANNEL_STANDARD_MODE;
hAudioInTopRightChannel.Init.Input.Pins = DFSDM_CHANNEL_FOLLOWING_CHANNEL_PINS;
/* Request to sample stable data for RIGHT micro on Falling edge */
hAudioInTopRightChannel.Init.Seriableface.Type = DFSDM_CHANNEL_SPI_FALLING;
hAudioInTopRightChannel.Init.SerialInterface.SpiClock = DFSDM_CHANNEL_SPI_CLOCK_INTERNAL;
hAudioInTopRightChannel.Init.Awd.FilterOrder = DFSDM_CHANNEL_FASTSINC_ORDER;
hAudioInTopRightChannel.Init.Awd.Oversampling = 10;
hAudioInTopRightChannel.Init.Offset = 0;
hAudioInTopRightChannel.Init.RightBitShift = DFSDM_RIGHT_BIT_SHIFT(AudioFreq);
if (HAL_OK != HAL_DFSDM_ChannelInit(&hAudioInTopRightChannel))
{
    return AUDIO_ERROR;
}

if (AudioIn_ChannelNumber > 2)
{
    /* CHANNEL 5 configuration */
    __HAL_DFSDM_CHANNEL_RESET_HANDLE_STATE(&hAudioInBottomLeftChannel);
    hAudioInBottomLeftChannel.Instance = DFSDM1_Channel5;
    hAudioInBottomLeftChannel.Init.OutputClock.Activation = ENABLE;
    hAudioInBottomLeftChannel.Init.OutputClock.Selection = DFSDM_CHANNEL_OUTPUT_CLOCK_AUDIO;
    /* Set the DFSDM clock OUT audio frequency configuration */
    hAudioInBottomLeftChannel.Init.OutputClock.Divider = DFSDM_CLOCK_DIVIDER(AudioFreq);
    hAudioInBottomLeftChannel.Init.Input.Multiplexer = DFSDM_CHANNEL_EXTERNAL_INPUTS;
    hAudioInBottomLeftChannel.Init.Input.DataPacking = DFSDM_CHANNEL_STANDARD_MODE;
    hAudioInBottomLeftChannel.Init.SerialInt

erface.Type = DFSDM_CHANNEL_SPI_RISING;
01796 hAudioInBottomLeftChannel.Init.SerialInt erface.SpiClock = DFSDM_CHANNEL_SPI_CLOCK_INTERNAL;
01797 hAudioInBottomLeftChannel.Init.Awd.Filte rOrder = DFSDM_CHANNEL_FASTSINC_ORDER;
01798 hAudioInBottomLeftChannel.Init.Awd.Oversampling = 10;
01799 hAudioInBottomLeftChannel.Init.Offset = 0;
01800 hAudioInBottomLeftChannel.Init.RightBitShift = DFSDM_RIGHT_BIT_SHIFT(AudioFreq);
01801 if(HAL_OK != HAL_DFSDM_ChannelInit(&hAudioInBottomLeftChannel))
01802 {
01803     return AUDIO_ERROR;
01804 }
01805
01806 /* CHANNEL 4 configuration * /
01807 __HAL_DFSDM_CHANNEL_RESET_HANDLE_STATE(& hAudioInBottomRightChannel);
01808 hAudioInBottomRightChannel.Instance = DFSDM1_Channel4;
01809 hAudioInBottomRightChannel.Init.OutputCl ock.Activation = ENABLE;
01810 hAudioInBottomRightChannel.Init.OutputCl ock.Selection = DFSDM_CHANNEL_OUTPUT_CLOCK_AUDIO;
01811 /* Set the DFSDM clock OUT audio frequen cy configuration * /
01812 hAudioInBottomRightChannel.Init.OutputCl ock.Divider = DFSDM_CLOCK_DIVIDER(AudioFreq);
01813 hAudioInBottomRightChannel.Init.Input.Mu ltiplexer = DFSDM_CHANNEL_EXTERNAL_INPUTS;
01814 hAudioInBottomRightChannel.Init.Input.Da taPacking = DFSDM_CHANNEL_STANDARD_MODE;
01815  hAudioInButtomRightChannel.Init.Input.Pins = DFSDM_CHANNEL_FOLLOWING_CHANNEL_PINS;
01816  /* Request to sample stable data for RIG HT micro on Falling edge */
01817  hAudioInButtomRightChannel.Init.SerialInterface.Type = DFSDM_CHANNEL_SPI_FALLING;
01818  hAudioInButtomRightChannel.Init.SerialInterface.SpiClock = DFSDM_CHANNEL_SPI_CLOCK_INTERNAL;
01819  hAudioInButtomRightChannel.Init.Awd.FilterOrder = DFSDM_CHANNEL_FASTSINC_ORDER;
01820  hAudioInButtomRightChannel.Init.Awd.Over sampling = 10;
01821  hAudioInButtomRightChannel.Init.Offset = 0;
01822  hAudioInButtomRightChannel.Init.RightBit Shift = DFSDM_RIGHT_BIT_SHIFT(AudioFreq);
01823  if(HAL_OK != HAL_DFSDM_ChannelInit(&hAudioInButtomRightChannel))
01824      {
01825         return AUDIO_ERROR;
01826      }
01827  }
01828 }************************************************************************************
01829  /************************** Filters configuration ******************************/
01830  /***********************************************************************************/
01831  /* FILTER 0 configuration */
01832  __HAL_DFSDM_FILTER_RESET_HANDLE_STATE(&hAudioInTopLeftFilter);
01833  hAudioInTopLeftFilter.Instance = AUDIO_DFSDMX_TOP_LEFT_FILTER;
hAudioInTopLeftFilter.Init.RegularParam.Trigger = DFSDM_FILTER_SW_TRIGGER;

hAudioInTopLeftFilter.Init.RegularParam.FastMode = ENABLE;

hAudioInTopLeftFilter.Init.RegularParam.DmaMode = ENABLE;

hAudioInTopLeftFilter.Init.InjectedParam.Trigger = DFSDM_FILTER_SW_TRIGGER;

hAudioInTopLeftFilter.Init.InjectedParam.ScanMode = ENABLE;

hAudioInTopLeftFilter.Init.InjectedParam.DmaMode = DISABLE;

hAudioInTopLeftFilter.Init.InjectedParam.ExtTrigger = DFSDM_FILTER_EXT_TRIG_TIM1_TRGO;

hAudioInTopLeftFilter.Init.InjectedParam.ExtTriggerEdge = DFSDM_FILTER_EXT_TRIG_RISING_EDGE;

hAudioInTopLeftFilter.Init.FilterParam.SinOrder = DFSDM_FILTER_ORDER(AudioFreq);

/* Set the DFSDM Filters Oversampling to have correct sample rate */

hAudioInTopLeftFilter.Init.FilterParam.Oversampling = DFSDM_OVER_SAMPLING(AudioFreq);

hAudioInTopLeftFilter.Init.FilterParam.IntOversampling = 1;

if(HAL_OK != HAL_DFSDM_FilterInit(&hAudioInTopLeftFilter))
{
    return AUDIO_ERROR;
}

/* Configure injected channel */

if(HAL_OK != HAL_DFSDM_FilterConfigRegChannel(&hAudioInTopLeftFilter, AUDIO_DFSDMx_TOP_LEFT_CHANNEL, DFSDM_CONTINUOUS_CONV_ON))
{
    return AUDIO_ERROR;
}
01857    /* FILTER 1 configuration */
01858    __HAL_DFSDM_FILTER_RESET_HANDLE_STATE(&hAudioInTopRightFilter);
01859    hAudioInTopRightFilter.Instance
01860        = AUDIO_DFSDMx_TOP_RIGHT_FILTER;
01861    hAudioInTopRightFilter.Init.RegularParam.Trigger
01862        = DFSDM_FILTER_SYNC_TRIGGER;
01863    hAudioInTopRightFilter.Init.RegularParam.FastMode
01864        = ENABLE;
01865    hAudioInTopRightFilter.Init.InjectedParam.
01866        Trigger
01867        = DFSDM_FILTER_SW_TRIGGER;
01868    hAudioInTopRightFilter.Init.InjectedParam.
01869        ScanMode
01870        = DISABLE;
01871    hAudioInTopRightFilter.Init.InjectedParam.
01872        DmaMode
01873        = DISABLE;
01874    hAudioInTopRightFilter.Init.InjectedParam.
01875        ExtTrigger
01876        = DFSDM_FILTER_EXT_TRIG_TIM1_TRGO;
01877    hAudioInTopRightFilter.Init.InjectedParam.
01878        ExtTriggerEdge
01879        = DFSDM_FILTER_EXT_TRIG_RISING_EDGE;
01880    hAudioInTopRightFilter.Init.FilterParam.
01881        SyncOrder
01882        = DFSDM_FILTER_ORDER(AudioFreq);
01883    /* Set the DFSDM Filters Oversampling to have correct sample rate */
01884    hAudioInTopRightFilter.Init.FilterParam.Oversampling
01885        = DFSDM_OVER_SAMPLING(AudioFreq);
01886    hAudioInTopRightFilter.Init.FilterParam.In
01887        tOversampling = 1;
01888    if(HAL_OK != HAL_DFSDM_FilterInit(&hAudioInTopRightFilter))
01889    {
01890        return AUDIO_ERROR;
01891    }
01892    /* Configure injected channel */
if(HAL_OK != HAL_DFSDM_FilterConfigRegChannel(&hAudioInTopRightFilter, AUDIO_DFSDMx_TOP_RIGHT_CHANNEL, DFSDM_CONTINUOUS_CONV_ON))
{
    return AUDIO_ERROR;
}

if(AudioIn_ChannelNumber > 2)
{
    /* FILTER 2 configuration */
    __HAL_DFSDM_FILTER_RESET_HANDLE_STATE(&hAudioInButtomLeftFilter);
    hAudioInButtomLeftFilter.Instance
        = AUDIO_DFSDMx_BUTTOM_LEFT_FILTER;
    hAudioInButtomLeftFilter.Init.RegularParam.Trigger
        = DFSDM_FILTER_SYNC_TRIGGER;
    hAudioInButtomLeftFilter.Init.RegularParam.FastMode
        = ENABLE;
    hAudioInButtomLeftFilter.Init.RegularParam.DmaMode
        = ENABLE;
    hAudioInButtomLeftFilter.Init.InjectedParam.Trigger
        = DFSDM_FILTER_SW_TRIGGER;
    hAudioInButtomLeftFilter.Init.InjectedParam.ScanMode
        = ENABLE;
    hAudioInButtomLeftFilter.Init.InjectedParam.DmaMode
        = DISABLE;
    hAudioInButtomLeftFilter.Init.InjectedParam.ExtTrigger
        = DFSDM_FILTER_EXT_TRIG_TIM1_TRGO;
    hAudioInButtomLeftFilter.Init.InjectedParam.ExtTriggerEdge
        = DFSDM_FILTER_EXT_TRIG_RISING_EDGE;
    hAudioInButtomLeftFilter.Init.FilterParam.SincOrder
        = DFSDM_FILTER_ORDER(AudioFreq);
    /* Set the DFSDM Filters Oversampling to
have correct sample rate */

```c
01898  hAudioInButtomLeftFilter.Init.FilterPara
01899  m.Oversampling = DFSDM_OVER_SAMPLING(AudioFreq);
01900  if(HAL_OK != HAL_DFSDM_FilterInit(&hAudio
01901         InButtomLeftFilter))
01902      { return AUDIO_ERROR;
01903  }
01904
01905  /* Configure injected channel */
01906  if(HAL_OK != HAL_DFSDM_FilterConfigRegCh
01907      annel(&hAudioInButtomLeftFilter, AUDIO_DFSDMx_BUTT
01908      OM_LEFT_CHANNEL, DFSDM_CONTINUOUS_CONV_ON))
01909      { return AUDIO_ERROR;
01910  }
01911  /* FILTER 3 configuration */
01912  __HAL_DFSDM_FILTER_RESET_HANDLE_STATE(&
01913         hAudioInButtomRightFilter);
01914  hAudioInButtomRightFilter.Instance
01915         = AUDIO_DFSDMx_BUTTOM_RIGHT_FI
01916  LTER;
01917  hAudioInButtomRightFilter.Init.RegularPa
01918  ram.Trigger = DFSDM_FILTER_SYNC_TRIGGER;
01919  hAudioInButtomRightFilter.Init.RegularPa
01920  ram.FastMode = ENABLE;
01921  hAudioInButtomRightFilter.Init.RegularPa
01922  ram.DmaMode = ENABLE;
01923  hAudioInButtomRightFilter.Init.InjectedP
01924  aram.Trigger = DFSDM_FILTER_SW_TRIGGER;
01925  hAudioInButtomRightFilter.Init.InjectedP
01926  aram.ScanMode = DISABLE;
01927  hAudioInButtomRightFilter.Init.InjectedP
```
aram.DmaMode = DISABLE;

if(HAL_OK != HAL_DFSDM_FilterInit(&hAudioInButtomRightFilter))
{
    return AUDIO_ERROR;
}

if(HAL_OK != HAL_DFSDM_FilterConfigRegChannel(&hAudioInButtomRightFilter, AUDIO_DFSDMx_BUTTOM_RIGHT_CHANNEL, DFSDM_CONTINUOUS_CONV_ON))
{
    return AUDIO_ERROR;
}

return AUDIO_OK;

/**
 * @brief De-initialize the Digital Filter for Sigma-Delta Modulators interface (DFSDM).
 * @retval AUDIO_OK if correct communicatio
static uint8_t DFSDMx_DeInit(void)
{
    /* De-initializes the DFSDM filters to allow access to DFSDM internal registers */
    if(HAL_OK != HAL_DFSDM_FilterDeInit(&hAudioInTopLeftFilter))
    {
        return AUDIO_ERROR;
    }
    if(HAL_OK != HAL_DFSDM_FilterDeInit(&hAudioInTopRightFilter))
    {
        return AUDIO_ERROR;
    }
    if(HAL_OK != HAL_DFSDM_ChannelDeInit(&hAudioInTopLeftChannel))
    {
        return AUDIO_ERROR;
    }
    if(HAL_OK != HAL_DFSDM_ChannelDeInit(&hAudioInTopRightChannel))
    {
        return AUDIO_ERROR;
    }
    if(AudioIn_ChannelNumber > 2)
    {
        /* De-initializes the DFSDM filters to allow access to DFSDM internal registers */
        if(HAL_OK != HAL_DFSDM_FilterDeInit(&hAu
if(HAL_OK != HAL_DFSMD_ChannelDeInit(&hAudioInButtomLeftChannel))
    {
        return AUDIO_ERROR;
    }

if(HAL_OK != HAL_DFSMD_ChannelDeInit(&hAudioInButtomRightChannel))
    {
        return AUDIO_ERROR;
    }

/* De-initializes the DFSDM channels to allow access to DFSDM internal registers */

/**
 * @brief Initialize the DFSDM channel MSP.
 *
 * @retval None
 */

static void DFSDMx_ChannelMspInit(void) 
{
    GPIO_InitTypeDef GPIO_InitStruct;
/* Enable DFSDM clock */
AUDIO_DFSDMx_CLK_ENABLE();

/* Enable GPIO clock */
AUDIO_DFSDMx_DMIC_DATIN_GPIO_CLK_ENABLE();
AUDIO_DFSDMx_CKOUT_DMIC_GPIO_CLK_ENABLE();

/* DFSDM pins configuration: DFSDM_CKOUT, DMIC_DATIN1 pins ------------------*/
GPIO_InitStruct.Pin = AUDIO_DFSDMx_CKOUT_PIN;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
HAL_GPIO_Init(AUDIO_DFSDMx_CKOUT_DMIC_GPIO_PORT, &GPIO_InitStruct);

/* DFSDM pin configuration: DMIC_DATIN1 pin -------------------------------*/
GPIO_InitStruct.Pin = AUDIO_DFSDMx_DMIC_DATIN1_PIN;
GPIO_InitStruct.Alternate = AUDIO_DFSDMx_DMIC_DATIN_AF;
HAL_GPIO_Init(AUDIO_DFSDMx_DMIC_DATIN_GPIO_PORT, &GPIO_InitStruct);

if (AudioIn_ChannelNumber > 2) {
    /* DFSDM pin configuration: DMIC_DATIN5 pin -------------------------------*/
    GPIO_InitStruct.Pin = AUDIO_DFSDMx_DMIC_DATIN5_PIN;
    GPIO_InitStruct.Alternate = AUDIO_DFSDMx_DMIC_DATIN5_AF;
    HAL_GPIO_Init(AUDIO_DFSDMx_DMIC_DATIN5_GPIO_PORT, &GPIO_InitStruct);
}
DMIC_DATIN_AF;
02028   HAL_GPIO_Init(AUDIO_DFSMDx_DMIC_DATIN_GPIO_PORT, &GPIO_InitStruct);
02029 }  
02030 }  
02031  /**
02032   * @brief DeInitialize the DFSDM channel M SP.
02033   * @retval None
02034 */
02035  
02036 static void DFSDMx_ChannelMspDeInit(void)  
02037 {  
02038   GPIO_InitTypeDef GPIO_InitStruct;  
02039  
02040   /* DFSDM pin configuration: DMIC_DATIN1 pin -------------------------------*/
02041   GPIO_InitStruct.Pin = AUDIO_DFSMDx_CKOUT_PIN;  
02042   HAL_GPIO_DeInit(AUDIO_DFSMDx_CKOUT_DMIC_GPIO_PORT, GPIO_InitStruct.Pin);  
02043   GPIO_InitStruct.Pin = AUDIO_DFSMDx_DMIC_DATIN1_PIN;  
02044   HAL_GPIO_DeInit(AUDIO_DFSMDx_DMIC_DATIN_GPIO_PORT, GPIO_InitStruct.Pin);  
02045  
02046   if(AudioIn_ChannelNumber > 2)  
02047   {  
02048     /* DFSDM pin configuration: DMIC_DATIN5 pin -------------------------------*/
02049     GPIO_InitStruct.Pin = AUDIO_DFSMDx_CKOUT_PIN;  
02050     HAL_GPIO_DeInit(AUDIO_DFSMDx_CKOUT_DMIC_GPIO_PORT, GPIO_InitStruct.Pin);  
02051     GPIO_InitStruct.Pin = AUDIO_DFSMDx_DMIC_DATIN5_PIN;  
02052     HAL_GPIO_DeInit(AUDIO_DFSMDx_DMIC_DATIN_
GPIO_PORT, GPIO_InitStruct.Pin);

02053  }
02054 }
02055
02056 /**
02057    * @brief Initialize the DFSDM filter MSP.
02058    * @retval None
02059 */

02060 static void DFSDMx_FilterMspInit(void)
02061 {
02062  /* Enable DFSDM clock */
02063  AUDIO_DFSDMx_CLK_ENABLE();
02064
02065  /* Enable the DMA clock */
02066  AUDIO_DFSDMx_DMAx_CLK_ENABLE();
02067
02068  /*********** Configure DMA stream for TOP LEFT microphone ***********/
02069  hDmaTopLeft.Init.Direction = DMA
      _PERIPH_TO_MEMORY;
02070  hDmaTopLeft.Init.PeriphInc = DMA
      _PINC_DISABLE;
02071  hDmaTopLeft.Init.MemInc = DMA
      _MINC_ENABLE;
02072  hDmaTopLeft.Init.PeriphDataAlignment = AUD
      IO_DFSDMx_DMAx_PERIPH_DATA_SIZE;
02073  hDmaTopLeft.Init.MemDataAlignment = AUD
      IO_DFSDMx_DMAx_MEM_DATA_SIZE;
02074  hDmaTopLeft.Init.Mode = DMA
      _CIRCULAR;
02075  hDmaTopLeft.Init.Priority = DMA
      _PRIORITY_HIGH;
02076  hDmaTopLeft.Instance = AUD
      IO_DFSDMx_DMAx_TOP_LEFT_STREAM;
02077  hDmaTopLeft.Init.Channel = AUD
      IO_DFSDMx_DMAx_CHANNEL;
02078
/* Associate the DMA handle */
__HAL_LINKDMA(&hAudioInTopLeftFilter, hdmaReg, hDmaTopLeft);

/* Reset DMA handle state */
__HAL_DMA_RESET_HANDLE_STATE(&hDmaTopLeft);

/* Configure the DMA Channel */
HAL_DMA_Init(&hDmaTopLeft);

/* DMA IRQ Channel configuration */
HAL_NVIC_SetPriority(AUDIO_DFSDMx_DMAx_TOP_LEFT_IRQ, AUDIO_OUT_IRQ_PREPRIO, 0);
HAL_NVIC_EnableIRQ(AUDIO_DFSDMx_DMAx_TOP_LEFT_IRQ);

/*********** Configure DMA stream for TOP RIGHT microphone *************/
hDmaTopRight.Init.Direction = DMA_PERIPH_TO_MEMORY;
hDmaTopRight.Init.PeriphInc = DMA_PINC_DISABLE;
hDmaTopRight.Init.MemInc = DMA_MINC_ENABLE;
hDmaTopRight.Init.PeriphDataAlignment = AUDIO_DFSDMx_DMAx_PERIPH_DATA_SIZE;
hDmaTopRight.Init.MemDataAlignment = AUDIO_DFSDMx_DMAx_MEM_DATA_SIZE;
hDmaTopRight.Init.Mode = DMA_CIRCULAR;
hDmaTopRight.Init.Priority = DMA_PRIORITY_HIGH;
hDmaTopRight.Instance = AUDIO_DFSDMx_DMAx_TOP_RIGHT_STREAM;
hDmaTopRight.Init.Channel = AUDIO_DFSDMx_DMAx_TOP_RIGHT_CHANNEL;
DIO_DFSDMx_DMAx_CHANNEL;
02103 /* Associate the DMA handle */
02104 __HAL_LINKDMA(&hAudioInTopRightFilter, hdmaReg, hDmaTopRight);
02106 /* Reset DMA handle state */
02107 __HAL_DMA_RESET_HANDLE_STATE(&hDmaTopRight);
02109 /* Configure the DMA Channel */
02110 HAL_DMA_Init(&hDmaTopRight);
02112 /* DMA IRQ Channel configuration */
02113 HAL_NVIC_SetPriority(AUDIO_DFSDMx_DMAx_TOP_RIGHT_IRQ, AUDIO_OUT_IRQ_PREPRIO, 0);
02114 HAL_NVIC_EnableIRQ(AUDIO_DFSDMx_DMAx_TOP_RIGHT_IRQ);
02116 if(AudioIn_ChannelNumber > 2)
02117 { /************ Configure DMA stream for BOTTOM LEFT microphone *************/
02120   hDmaButtomLeft.Init.Direction = DMA_PERIPH_TO_MEMORY;
02121   hDmaButtomLeft.Init.PeriphInc = DMA_PINC_DISABLE;
02122   hDmaButtomLeft.Init.MemInc = DMA_MINC_ENABLE;
02123   hDmaButtomLeft.Init.PeriphDataAlignment = AUDIO_DFSDMx_DMAx_PERIPH_DATA_SIZE;
02124   hDmaButtomLeft.Init.MemDataAlignment = AUDIO_DFSDMx_DMAx_MEM_DATA_SIZE;
02125   hDmaButtomLeft.Init.Mode = DMA_CIRCULAR;
02126   hDmaButtomLeft.Init.Priority = DMA_PRIORITY_HIGH;
hDmaButtomLeft.Instance = AUDIO_DFSDMx_DMAx_BUTTOM_LEFT_STREAM;
hDmaButtomLeft.Init.Channel = AUDIO_DFSDMx_DMAx_CHANNEL;

/* Associate the DMA handle */
__HAL_LINKDMA(&hAudioInButtomLeftFilter, hdmaReg, hDmaButtomLeft);

/* Reset DMA handle state */
__HAL_DMA_RESET_HANDLE_STATE(&hDmaButtomLeft);

/* Configure the DMA Channel */
HAL_DMA_Init(&hDmaButtomLeft);

/* DMA IRQ Channel configuration */
HAL_NVIC_SetPriority(AUDIO_DFSDMx_DMAx_BUTTOM_LEFT_IRQ, AUDIO_OUT_IRQ_PREPRIO, 0);
HAL_NVIC_EnableIRQ(AUDIO_DFSDMx_DMAx_BUTTOM_LEFT_IRQ);

/*********** Configure DMA stream for BUTTOM RIGHT microphone *************/

hDmaButtomRight.Init.Direction = DMA_PERIPH_TO_MEMORY;
hDmaButtomRight.Init.PeriphInc = DMA_PINC_DISABLE;
hDmaButtomRight.Init.MemInc = DMA_MINC_ENABLE;
hDmaButtomRight.Init.PeriphDataAlignment = AUDIO_DFSDMx_DMAx_PERIPH_DATA_SIZE;
hDmaButtomRight.Init.MemDataAlignment = AUDIO_DFSDMx_DMAx_MEM_DATA_SIZE;
hDmaButtomRight.Init.Mode = DMA_CIRCULAR;
02151       hDmaButtomRight.Init.Priority
02152           = DMA_PRIORITY_HIGH;
02153       hDmaButtomRight.Instance
02154           = AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_STREAM;
02155       hDmaButtomRight.Init.Channel
02156           = AUDIO_DFSDMx_DMAx_CHANNEL;
02157
02158        /* Associate the DMA handle */
02159        __HAL_LINKDMA(&hAudioInButtomRightFilter,
02160            hdmaReg, hDmaButtomRight);
02161
02162        /* Reset DMA handle state */
02163        __HAL_DMA_RESET_HANDLE_STATE(&hDmaButtomRight);
02164
02165        /* Configure the DMA Channel */
02166        HAL_DMA_Init(&hDmaButtomRight);
02167
02168        /* DMA IRQ Channel configuration */
02169        HAL_NVIC_SetPriority(AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_IRQ, AUDIO_OUT_IRQ_PREPRIO, 0);
02170        HAL_NVIC_EnableIRQ(AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_IRQ);
02171      }
02172  }
02173
02174 static void DFSDMx_FilterMspDeInit(void)
02175  {
02176        /* Configure the DMA Channel */
02177        HAL_DMA_DeInit(&hDmaTopLeft);
02178        HAL_DMA_DeInit(&hDmaTopRight);
02179        if(AudioIn_ChannelNumber > 2)
02180              /*
HAL_DMA_DeInit(&hDmaButtonLeft);
HAL_DMA_DeInit(&hDmaButtonRight);
}
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td>Modules</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### STM32F769I_DISCOVERY AUDIO

STM32F769I_DISCOVERY

This file includes the low layer driver for wm8994 Audio Codec available on STM32F769I-DISCOVERY discoveryuation board(MB1225). More...
# Modules

<table>
<thead>
<tr>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F769I_DISCOVERY_AUDIO Private Types</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY_AUDIO Private Defines</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY_AUDIO Private Macros</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY_AUDIO Private Variables</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY_AUDIO Private Function Prototypes</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY_AUDIO_Out Private Function Prototypes</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY_AUDIO_In Private Functions</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY_AUDIO_Exported Types</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY_AUDIO_Exported Constants</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY_AUDIO_Exported Macros</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY_AUDIO_OUT_Exported Functions</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY_AUDIO_IN_Exported Functions</td>
</tr>
</tbody>
</table>
Detailed Description

This file includes the low layer driver for wm8994 Audio Codec available on STM32F769I-DISCOVERY discoveryuation board(MB1225).
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**STM32F769I_DISCOVERY LCD**

STM32F769I_DISCOVERY
## Modules

<table>
<thead>
<tr>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCD Private Defines</td>
</tr>
<tr>
<td>LCD Private TypesDefinitions</td>
</tr>
<tr>
<td>LCD Private Macros</td>
</tr>
<tr>
<td>STM32F769I DISCOVERY LCD Exported Variables</td>
</tr>
<tr>
<td>LCD Private Variables</td>
</tr>
<tr>
<td>LCD Private FunctionPrototypes</td>
</tr>
<tr>
<td>LCD Exported Functions</td>
</tr>
<tr>
<td>STM32F769I DISCOVERY LCD Exported Constants</td>
</tr>
<tr>
<td>STM32F769I DISCOVERY LCD Exported Types</td>
</tr>
<tr>
<td>STM32F769I DISCOVERY LCD Exported Macro</td>
</tr>
</tbody>
</table>

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery
BSP User Manual by [doxygen](https://www.doxygen.org/) 1.7.6.1
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td>Data Structures</td>
<td>Typedefs</td>
</tr>
</tbody>
</table>

**STM32F769I DISCOVERY LCD Exported Types**

STM32F769I_DISCOVERY_LCD
## Data Structures

<table>
<thead>
<tr>
<th>struct</th>
<th>LCD_DrawPropTypeDef</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LCD Drawing main properties. More...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>struct</th>
<th>Point</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LCD Drawing point (pixel) geometric definition. More...</td>
</tr>
</tbody>
</table>
## Typedefs

typedef Point * pPoint

Pointer on LCD Drawing point (pixel) geometric definition.
### Enumerations

<table>
<thead>
<tr>
<th>Enum Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Text_AlignModeTypdef</strong></td>
<td><code>enum Text_AlignModeTypdef { CENTER_MODE = 0x01, RIGHT_MODE = 0x02, LEFT_MODE = 0x03 }</code></td>
</tr>
<tr>
<td><strong>LCD_OrientationTypeDef</strong></td>
<td><code>enum LCD_OrientationTypeDef { LCD_ORIENTATION_PORTRAIT = 0x00, LCD_ORIENTATION_LANDSCAPE = 0x01, LCD_ORIENTATION_INVALID = 0x02 }</code></td>
</tr>
</tbody>
</table>

LCD drawing Line alignment mode definitions. More...

LCD_OrientationTypeDef Possible values of Display Orientation. More...
typedef Point* pPoint

Pointer on LCD Drawing point (pixel) geometric definition.

Definition at line 287 of file stm32f769i_discovery_lcd.h.
**Enumeration Type Documentation**

**enum** LCD_OrientationTypeDef

LCD_OrientationTypeDef Possible values of Display Orientation.

**Enumerator:**

- **LCD_ORIENTATION_PORTRAIT** Portrait orientation choice of LCD screen
- **LCD_ORIENTATION_LANDSCAPE** Landscape orientation choice of LCD screen
- **LCD_ORIENTATION_INVALID** Invalid orientation choice of LCD screen

Definition at line 305 of file stm32f769i_discovery_lcd.h.

**enum** Text_AlignModeTypdef

LCD drawing Line alignment mode definitions.

**Enumerator:**

- **CENTER_MODE** Center mode
- **RIGHT_MODE** Right mode
- **LEFT_MODE** Left mode
Definition at line 292 of file stm32f769i_discovery_lcd.h.
stm32f769i_discovery_lcd.h

Go to the documentation of this file.

```c
/**
 * @file    stm32f769i_discovery_lcd.h
 * @author  MCD Application Team
 * @version V2.0.0
 * @date    30-December-2016
 * @brief   This file contains the common defines and functions prototypes for
 *          the stm32469i_discovery_lcd.c driver.
 * @attention
 * <h2><center>© COPYRIGHT(c) 2016 STMicroelectronics</center></h2>
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
```
1. Redistributions of source code must retain the above copyright notice,
   this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.
3. Neither the name of STMicroelectronics nor the names of its contributors
   may be used to endorse or promote products derived from this software
   without specific prior written permission.

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

#ifndef __STM32F769I_DISCOVERY_LCD_H
#define __STM32F769I_DISCOVERY_LCD_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes -----------------------------------------------*/
/* Include LCD component Driver */

/* Include OTM8009A LCD Driver IC code */
#include "../Components/otm8009a/otm8009a.h"

/* Include ADV7533 HDMI Driver IC code */
#include "../Components/adv7533/adv7533.h"

/* Include SDRAM Driver */
#include "stm32f769i_discovery_sdram.h"
#include "stm32f769i_discovery.h"

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

#include <string.h> /* use of memset() */

/** @addtogroup BSP */
* @{
*/
/**
 * @addtogroup STM32F769I_DISCOVERY
 *
 */

/**
 * @addtogroup STM32F769I_DISCOVERY_LCD
 *
 */

/**
 * @defgroup STM32F769I_DISCOVERY_LCD_Exported_Constants STM32F769I_DISCOVERY_LCD_Exported_Constants
 *
 */

#define BSP_LCD_DMA2D_IRQHandler DMA2D_IRQHandler
#define BSP_LCD_DSI_IRQHandler DSI_IRQHandler
#define BSP_LCD_LTDC_IRQHandler LTDC_IRQHandler
#define BSP_LCD_LTDC_ER_IRQHandler LTDC_ER_IRQHandler
#define LCD_LayerCfgTypeDef LTDC_LayerCfgTypeDef

#define LCD_FB_START_ADDRESS ((uint32_t)0xC0000000)

#define LTDC_MAX_LAYER_NUMBER ((uint32_t)2)

#define LCD_FB_START_ADDRESS 0x00000000

/**
 * @brief LCD FB_StartAddress
 */

/**
 * @brief Maximum number of LTDC layers
 */

/**
 * @brief LTDC Background layer index
 */
```c
/*
 * 
 * define LTDC_ACTIVE_LAYER_BACKGROUND ((uint32_t) 0)
 */

/** @brief LTDC Foreground layer index */
#define LTDC_ACTIVE_LAYER_FOREGROUND ((uint32_t) 1)

/** @brief Number of LTDC layers */
#define LTDC_NB_OF_LAYERS ((uint32_t) 2)

/** @brief LTDC Default used layer index */
#define LTDC_DEFAULT_ACTIVE_LAYER LTDC_ACTIVE_LAYER_FOREGROUND

/** @brief LCD status structure definition */
#define LCD_OK 0x00
#define LCD_ERROR 0x01
#define LCD_TIMEOUT 0x02

/** @brief LCD Display OTM8009A DSI Virtual Channel ID */
#define LCD_OTM8009A_ID ((uint32_t) 0)

/** @brief HDMI ADV7533 DSI Virtual Channel ID */
```
```c
#define HDMI_ADV7533_ID ((uint32_t) 0) /* HDMI Format */
/**
 * @brief HDMI Foramt
 */
#define HDMI_FORMAT_720_480 ((uint8_t) 0x00) /* 720_480 format choice of HDMI display */
#define HDMI_FORMAT_720_576 ((uint8_t) 0x01) /* 720_576 format choice of HDMI display */
/**
 * @brief LCD color definitions values in ARGB8888 format.
 */
/**
 * @brief Blue value in ARGB8888 format
 */
#define LCD_COLOR_BLUE ((uint32_t) 0xFF0000FF)
/**
 * @brief Green value in ARGB8888 format
 */
#define LCD_COLOR_GREEN ((uint32_t) 0xFF00FF00)
/**
 * @brief Red value in ARGB8888 format
 */
#define LCD_COLOR_RED ((uint32_t) 0xFFFF0000)
/**
 * @brief Cyan value in ARGB8888 format
 */
#define LCD_COLOR_CYAN ((uint32_t) 0xFF00FFFF)
/**
 * @brief Magenta value in ARGB8888 format
 */
#define LCD_COLOR_MAGENTA ((uint32_t) 0xFF00FFFF)
```
```c
#define LCD_COLOR_MAGENTA (uint32_t) 0xFFFF00FF
/**	@brief Yellow value in ARGB8888 format
*/
#define LCD_COLOR_YELLOW (uint32_t) 0xFFFFFF00
/**	@brief Light Blue value in ARGB8888 format
*/
#define LCD_COLOR_LIGHTBLUE (uint32_t) 0xFF8080FF
/**	@brief Light Green value in ARGB8888 format
*/
#define LCD_COLOR_LIGHTGREEN (uint32_t) 0xFF80FF80
/**	@brief Light Red value in ARGB8888 format
*/
#define LCD_COLOR_LIGHTRED (uint32_t) 0xFFFF8080
/**	@brief Light Cyan value in ARGB8888 format
*/
#define LCD_COLOR_LIGHTCYAN (uint32_t) 0xFF80FFFF
/**	@brief Light Magenta value in ARGB8888 format
*/
#define LCD_COLOR_LIGHTMAGENTA (uint32_t) 0xFFFF80FF
```
/**	@brief	Light	Yellow	value	in	ARGB8888	format
*/
#define	LCD_COLOR_LIGHTYELLOW			((uint32_t)
0xFFFFFF80)

/**	@brief	Dark	Blue	value	in	ARGB8888	format
*/
#define	LCD_COLOR_DARKBLUE						((uint32_t)
0xFF000080)

/**	@brief	Light	Dark	Green	value	in	ARGB8888	format
*/
#define	LCD_COLOR_DARKGREEN						((uint32_t)
0xFF008000)

/**	@brief	Light	Dark	Red	value	in	ARGB8888	format
*/
#define	LCD_COLOR_DARKRED						
#define	LCD_COLOR_DARKCYAN						((uint32_t)
0xFF008080)

/**	@brief	Dark	Cyan	value	in	ARGB8888	format
*/
#define	LCD_COLOR_DARKMAGENTA			((uint32_t)
0xFF800080)

/**	@brief	Dark	Magenta	value	in	ARGB8888	format
*/
#define	LCD_COLOR_DARKCYAN						((uint32_t)
0xFF008080)

/**	@brief	Dark	Magenta	value	in	ARGB8888	format
*/
#define	LCD_COLOR_DARKMAGENTA			((uint32_t)
0xFF800080)
/** @brief Dark Yellow value in ARGB8888 format */
#define LCD_COLOR_DARKYELLOW ((uint32_t) 0xFF808000)

/** @brief White value in ARGB8888 format */
#define LCD_COLOR_WHITE ((uint32_t) 0xFFFFFFFF)

/** @brief Light Gray value in ARGB8888 format */
#define LCD_COLOR_LIGHTGRAY ((uint32_t) 0xFFD3D3D3)

/** @brief Gray value in ARGB8888 format */
#define LCD_COLOR_GRAY ((uint32_t) 0xFF808080)

/** @brief Dark Gray value in ARGB8888 format */
#define LCD_COLOR_DARKGRAY ((uint32_t) 0xFF404040)

/** @brief Black value in ARGB8888 format */
#define LCD_COLOR_BLACK ((uint32_t) 0xFF000000)

/** @brief Brown value in ARGB8888 format */
#define LCD_COLOR_BROWN ((uint32_t) 0xFFA52A2A)
/**
 * @brief Orange value in ARGB8888 format
 */
#define LCD_COLOR_ORANGE ((uint32_t) 0xFFFFA500)

/**
 * @brief Transparent value in ARGB8888 format
 */
#define LCD_COLOR_TRANSPARENT ((uint32_t) 0xFF000000)

/**
 * @brief LCD default font
 */
#define LCD_DEFAULT_FONT Font24

/**
 * @brief Possible values of pixel data format (ie color coding) transmitted on DSI Data lane in DSI packets
 */
#define LCD_DSI_PIXEL_DATA_FMT_RBG888 DSI_RGB888 /*!< DSI packet pixel format chosen is RGB 888 : 24 bpp */
#define LCD_DSI_PIXEL_DATA_FMT_RBG565 DSI_RGB565 /*!< DSI packet pixel format chosen is RGB 565 : 16 bpp */
typedef struct {
    uint32_t TextColor; /*!< Specifies the color of text */
    uint32_t BackColor; /*!< Specifies the background color below the text */
    sFONT *pFont;      /*!< Specifies the font used for the text */
} LCD_DrawPropTypeDef;

typedef struct {
    int16_t X;    /*!< geometric X position of drawing */
    int16_t Y;    /*!< geometric Y position of drawing */
} Point;

typedef Point * pPoint;

/* @brief LCD drawing Line alignment mode */
typedef enum {
CENTER_MODE  = 0x01, /*!< Center mode */
RIGHT_MODE   = 0x02, /*!< Right mode */
LEFT_MODE    = 0x03 /*!< Left mode */
} Text_AlignMode_TypeDef;

/**
@brief LCD_OrientationTypeDef
Possible values of Display Orientation
*/
typedef enum {
LCD_ORIENTATION_PORTRAIT = 0x00, /*!< Portrait orientation choice of LCD screen */
LCD_ORIENTATION_LANDSCAPE = 0x01, /*!< Landscape orientation choice of LCD screen */
LCD_ORIENTATION_INVALID   = 0x02 /*!< Invalid orientation choice of LCD screen */
} LCD_Orientation_TypeDef;
/**
 * @addtogroup STM32F769I_DISCOVERY_LCD_Exported_Functions
 *
 */

uint8_t BSP_LCD_Init(void);
uint8_t BSP_LCD_InitEx(LCD_OrientationTypeDef orientation);
uint8_t BSP_LCD_HDMIInitEx(uint8_t format);

void BSP_LCD_MspDeInit(void);
void BSP_LCD_MspInit(void);
void BSP_LCD_Reset(void);

uint32_t BSP_LCD_GetXSize(void);
uint32_t BSP_LCD_GetYSize(void);

void BSP_LCD_SetXSize(uint32_t imageWidthPixels);
void BSP_LCD_SetYSize(uint32_t imageHeightPixels);

void BSP_LCD_LayerDefaultInit(uint16_t LayerIndex, uint32_t FB_Address);
void BSP_LCD_SetTransparency(uint32_t LayerIndex, uint8_t Transparency);
void BSP_LCD_SetLayerAddress(uint32_t LayerIndex, uint32_t Address);
void BSP_LCD_SetColorKeying(uint32_t LayerIndex, uint32_t RGBValue);
void BSP_LCD_ResetColorKeying(uint32_t LayerIndex);
void BSP_LCD_SetLayerWindow(uint16_t LayerIndex, uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height);

void BSP_LCD_SelectLayer(uint32_t LayerIndex);
void BSP_LCD_SetLayerVisible(uint32_t LayerIndex);
yerIndex, FunctionalState State);

void BSP_LCD_SetTextColor(uint32_t Color);
uint32_t BSP_LCD_GetTextColor(void);
void BSP_LCD_SetBackColor(uint32_t Color);
uint32_t BSP_LCD_GetBackColor(void);
void BSP_LCD_SetFont(sFONT *fonts);
uint32_t *BSP_LCD_GetFont(void);

void BSP_LCD_ReadPixel(uint16_t Xpos, uint16_t Ypos);
void BSP_LCD_DrawPixel(uint16_t Xpos, uint16_t Ypos, uint32_t pixel);
void BSP_LCD_Clear(uint32_t Color);
void BSP_LCD_ClearStringLine(uint32_t Line);
void BSP_LCD_DisplayStringAtLine(uint16_t Line, uint8_t *ptr);
void BSP_LCD_DisplayStringAt(uint16_t Xpos, uint16_t Ypos, uint8_t *Text, Text_AlignModeTypeDef Mode);
void BSP_LCD_DisplayChar(uint16_t Xpos, uint16_t Ypos, uint8_t Ascii);
void BSP_LCD_DrawHLine(uint16_t Xpos, uint16_t Ypos, uint16_t Length);
void BSP_LCD_DrawVLine(uint16_t Xpos, uint16_t Ypos, uint16_t Length);
void BSP_LCD_DrawLine(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2);
void BSP_LCD_DrawRect(uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height);
void BSP_LCD_DrawCircle(uint16_t Xpos, uint16_t Ypos, uint16_t Radius);
void BSP_LCD_DrawPolygon(pPoint Points,
uint16_t PointCount);
00367 void BSP_LCD_DrawEllipse(int Xpos, int Ypos, int XRadius, int YRadius);
00368 void BSP_LCD_DrawBitmap(uint32_t Xpos, uint32_t Ypos, uint8_t *pbmp);
00369
00370 void BSP_LCD_FillRect(uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height);
00371 void BSP_LCD_FillCircle(uint16_t Xpos, uint16_t Ypos, uint16_t Radius);
00372 void BSP_LCD_FillPolygon(pPoint Points, uint16_t PointCount);
00373 void BSP_LCD_FillEllipse(int Xpos, int Ypos, int XRadius, int YRadius);
00374
00375 void BSP_LCD_DisplayOn(void);
00376 void BSP_LCD_DisplayOff(void);
00377 void BSP_LCD_SetBrightness(uint8_t BrightnessValue);
00378 00379 /**<
00380  */
00381 */
00382
00383 /**< @defgroup STM32F769I_DISCOVERY_LCD_Exported_Variables STM32F769I DISCOVERY LCD Exported Variables */
00384 */
00385 */
00386
00387 /**< @brief DMA2D handle variable */
00388 extern DMA2D_HandleTypeDef hdma2d_discovery;
00389
00390 /**<
00391 */
00392 */
00393
#ifdef __cplusplus
}
#endif

/*	__STM32F769I_DISCOVERY_LCD_H	*/

/************************	(C)	COPYRIGHT	STMicroelectronics	*****END	OF	FILE****/
/**
 * @file stm32f769i_discovery_lcd.c
 * @author MCD Application Team
 * @version V2.0.0
 * @date 30-December-2016
 * @brief This file includes the driver for Liquid Crystal Display (LCD) module
 * mounted on STM32F769I-DISCOVERY board.
 *
 * @attention
 *
 * <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>
 *
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice,
   this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

3. Neither the name of STMicroelectronics nor the names of its contributors
   may be used to endorse or promote products derived from this software
   without specific prior written permission.

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

- This driver is used to drive directly in video mode a LCD TFT using the DSI interface.

The following IPs are implied: DSI Host IP block working in conjunction to the LTDC controller.

- This driver is linked by construction to LCD KoD mounted on board MB1166.

2. Driver description:

+ Initialization steps:
  - Initialize the LCD using the BSP_LCD_Init() function.
  - Select the LCD layer to be used using the BSP_LCD_SelectLayer() function.
  - Enable the LCD display using the BSP_LCD_DisplayOn() function.

+ Options
  - Configure and enable the color keying functionality using the BSP_LCD_SetColorKeying() function.
  - Modify in the fly the transparency and/or the frame buffer address using the following functions:
  - BSP_LCD_SetTransparency()
- BSP_LCD_SetLayerAddress()

+ Display on LCD
  o Clear the whole LCD using BSP_LCD_Clear() function or only one specified string line using the BSP_LCD_ClearStringLine() function.
  o Display a character on the specified line and column using the BSP_LCD_DisplayChar() function or a complete string line using the BSP_LCD_DisplayStringAtLine() function.
  o Display a string line on the specified position (x, y in pixel) and align mode using the BSP_LCD_DisplayStringAtLine() function.
  o Draw and fill a basic shapes (dot, line, rectangle, circle, ellipse, .. bitmap)

- Displays on the specified position (x, y in pixel) and align mode using the BSP_LCD_DisplayStringAtLine() function.

---

**Includes**

```c
#include "stm32f769i_discovery_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"
```

/** @addtogroup BSP */
* @{
  */

```java
* @addtogroup BSP */
```

*/

*/

*/
/** @addtogroup STM32F769I_DISCOVERY */

/**
 * @defgroup STM32F769I_DISCOVERY_LCD STM32F769I_DISCOVERY LCD
 */

/**
 * @defgroup STM32F769I_DISCOVERY_LCD_Private_Defines LCD Private Defines
 */

#if defined(USE_LCD_HDMI)
#define HDMI_ASPECT_RATIO_16_9 ADV7533_ASPECT_RATIO_16_9
#define HDMI_ASPECT_RATIO_4_3 ADV7533_ASPECT_RATIO_4_3
#endif /* USE_LCD_HDMI */

#define LCD_DSI_ID 0x11
#define LCD_DSI_ADDRESS TS_I2C_ADDRESS
#define LCD_DSI_ID_REG 0xA8

static DSI_VidCfgTypeDef hdsivideo_handle;

/** */

/** @defgroup STM32F769I_DISCOVERY_LCD_Private_TypesDefinitions LCD Private TypesDefinitions */

#if defined(USE_LCD_HDMI)
/** @brief DSI timming params used for different HDMI adapter */
typedef struct {
    uint16_t  HACT;
    uint16_t  HSYNC;
    uint16_t  HBP;
    uint16_t  HFP;
    uint16_t  VACT;
    uint16_t  VSYNC;
    uint16_t  VBP;
    uint16_t  VFP;
    uint8_t   ASPECT_RATIO;
    uint8_t   RGB_CODING;
} HDMI_FormatTypeDef;

typedef struct {
    uint16_t  NullPacketSize;
    uint16_t  NumberOfChunks;
    uint16_t  PacketSize;
} HDMI_DSIPacketTypeDef;

typedef struct {
    uint16_t  PLLSAIN;
    uint16_t  PLLSAIR;
    uint32_t  PCLK;
    uint16_t  IDF;
    uint16_t  NDIV;
} HDMI_PLLTypeDef;
uint16_t ODF;
uint16_t LaneByteClock;
uint16_t TXEscapeCkdiv;
}

# endif /* USE_LCD_HDMI */
/**
 * @}
 */

/**
 * @defgroup STM32F769I_DISCOVERY_LCD_Private_Macros LCD Private Macros
 */

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

#define POLY_X(Z) ((int32_t)((Points + (Z))->X))
#define POLY_Y(Z) ((int32_t)((Points + (Z))->Y))

/**
 * @}
 */

/**
 * @defgroup STM32F769I_DISCOVERY_LCD_Exported_Variables STM32F769I DISCOVERY LCD Exported Variables
 */

DMA2D_HandleTypeDef hdma2d_discovery;
LTDC_HandleTypeDef hltdc_discovery;
DSI_HandleTypeDef hdsi_discovery;
uint32_t lcd_x_size = OTM8009A_800X480_WIDTH;
uint32_t lcd_y_size = OTM8009A_800X480_HEIGHT;
/**
 * @defgroup STM32F769I_DISCOVERY_LCD_Private_Variables LCD Private Variables
 *
 */

#if defined(USE_LCD_HDMI)
/**
 * @brief DSI timing used for different HDMI resolution (720x480 and 720x576)
 */
HDMI_FormatTypeDef HDMI_Format[2] =
{
    /* HA HS HB HF VA VS VB VF ASPECT BPP */
    {720, 62, 60, 30, 480, 6, 19, 9, HDMI_ASPECT_RATIO_4_3, LCD_DSI_PIXEL_DATA_FMT_RBG888},
    {720, 64, 68, 12, 576, 5, 39, 5, HDMI_ASPECT_RATIO_16_9, LCD_DSI_PIXEL_DATA_FMT_RBG888}
};

/**
 * @brief DSI packet size used for different HDMI resolution (720x480 and 720x576)
 */
HDMI_DSIPacketTypeDef HDMI_DSIPacket[2] =
{
    /* NP NC VP */
    {0, 1, 720},
    {0, 1, 720}
};

/**
 * @brief LTDC PLL settings used for different HDMI resolution (720x480 and 720x576)
 */

HDMI_PLLConfigTypeDef HDMI_PLLConfig[4] =
{
  /* N  DIV Pclk  IDF  NDIV ODF  LBClk TXEscapeCkdiv */
  {325, 6, 27083, DSI_PLL_IN_DIV5, 65, DSI_PLL_OUT_DIV1, 40625, 3},
  {325, 6, 27083, DSI_PLL_IN_DIV5, 65, DSI_PLL_OUT_DIV1, 40625, 3}
};
#endif /* USE_LCD_HDMI */

/**
 * @brief Default Active LTDC Layer in which drawing is made is LTDC Layer Background
 */
static uint32_t ActiveLayer = LTDC_ACTIVE_LAYER_BACKGROUND;

/**
 * @brief Current Drawing Layer properties variable
 */
static LCD_DrawPropTypeDef DrawProp[LTDC_MAX_LAYER_NUMBER];

/**
 * @} */

/** @defgroup STM32F769I_DISCOVERY_LCD_Private_FunctionPrototypes LCD Private FunctionPrototypes */
* @} */
static void DrawChar(uint16_t Xpos, uint16_t Ypos, const uint8_t *c);
static void FillTriangle(uint16_t x1, uint16_t x2, uint16_t x3, uint16_t y1, uint16_t y2, uint16_t y3);
static void LL_FillBuffer(uint32_t LayerIndex, void *pDst, uint32_t xSize, uint32_t ySize, uint32_t OffLine, uint32_t ColorIndex);
static void LL_ConvertLineToARGB8888(void *pSrc, void *pDst, uint32_t xSize, uint32_t ColorMode);
static uint16_t LCD_IO_GetID(void);
/**
 * @}
 */

/** @defgroup STM32F769I_DISCOVERY_LCD_Exported_Functions LCD Exported Functions
 *
 */

/** @brief Initializes the DSI LCD.
 *
 */

uint8_t BSP_LCD_Init(void)
{
    return (BSP_LCD_InitEx(LCD_ORIENTATION_LANDSCAPE));
}

/**
 * @brief Initializes the DSI LCD.
 *
 */

uint8_t BSP_LCD_Init(void)
{
    return (BSP_LCD_InitEx(LCD_ORIENTATION_LANDSCAPE));
}

/**
 * @brief Initializes the DSI LCD.
 *
 */

uint8_t BSP_LCD_Init(void)
{
    return (BSP_LCD_InitEx(LCD_ORIENTATION_LANDSCAPE));
}

/**
 * @brief Initializes the DSI LCD.
 *
 */

uint8_t BSP_LCD_Init(void)
{
    return (BSP_LCD_InitEx(LCD_ORIENTATION_LANDSCAPE));
}
uint8_t BSP_LCD_InitEx(LCD_OrientationTypeDef orientation) {
  DSI_PLLInitTypeDef dsiPllInit;
  static RCC_PeriphCLKInitTypeDefPeriphClkInitStruct;
  uint32_t LcdClock = 27429; /*!< LcdClk = 27429 kHz */
  uint16_t read_id = 0;
  uint32_t laneByteClk_kHz = 0;
  uint32_t VSA; /*!< Vertical start active time in units of lines */
  uint32_t VBP; /*!< Vertical Back Porch time in units of lines */
  uint32_t VFP; /*!< Vertical Front Porch time in units of lines */
  uint32_t VACT; /*!< Vertical Active time in units of lines = imageSize Y in pixels to display */
  uint32_t HSA; /*!< Horizontal start active time in units of lcdClk */
  uint32_t HBP; /*!< Horizontal Back Porch time in units of lcdClk */
  uint32_t HFP; /*!< Horizontal Front Porch time in units of lcdClk */
  uint32_t HACT; /*!< Horizontal Active time in units of lcdClk = imageSize X in pixels to display */
}
/* Toggle Hardware Reset of the DSI LCD using its XRES signal (active low) */
BSP_LCD_Reset();

/* Check the connected monitor */
read_id = LCD_IO_GetID();

#if defined(USE_LCD_HDMI)
if (read_id == ADV7533_ID)
{
    return BSP_LCD_HDMIInitEx(HDMI_FORMAT_720_576);
}
else if (read_id != LCD_DSI_ID)
{
    return LCD_ERROR;
}
#else
if (read_id != LCD_DSI_ID)
{
    return LCD_ERROR;
}
#endif /* USE_LCD_HDMI */

/* Call first MSP Initialize only in case of first initialization
   * This will set IP blocks LTDC, DSI and DMA2D
   * - out of reset
   * - clocked
   * - NVIC IRQ related to IP blocks enabled
   */
BSP_LCD_MspInit();

/*****************************************************************************/

/*****************************************************************************/
/* Base address of DSI Host/Wrapper registers to be set before calling De-Init */
hdsi_discovery.Instance = DSI;

HAL_DSI_DeInit(&(hdsi_discovery));

dsiPllInit.PLLNDIV = 100;
dsiPllInit.PLLIDF = DSI_PLL_IN_DIV5;
dsiPllInit.PLLODF = DSI_PLL_OUT_DIV1;
laneByteClk_kHz = 62500; /* 500 MHz / 8 = 62.5 MHz = 62500 kHz */

/* Set number of Lanes */
hdsi_discovery.Init.NumberOfLanes = DSI_TW0_DATA_LANES;

/* TXEscapeCkdiv = f(LaneByteClk)/15.62 = 4 */
hdsi_discovery.Init.TXEscapeCkdiv = laneByteClk_kHz/15620;

HAL_DSI_Init(&(hdsi_discovery), &(dsiPllInit));

/* Timing parameters for all Video modes */
/* Set Timing parameters of LTDC depending on its chosen orientation */
if(orientation == LCD_ORIENTATION_PORTRAIT)
{
    lcd_x_size = OTM8009A_480X800_WIDTH; /* 480 */
    lcd_y_size = OTM8009A_480X800_HEIGHT; /* 800 */
}
else {
  /* lcd_orientation == LCD_ORIENTATION_LANDSCAPE */
  lcd_x_size = OTM8009A_800X480_WIDTH; /* 800 */
  lcd_y_size = OTM8009A_800X480_HEIGHT; /* 480 */
}

HACT = lcd_x_size;
VACT = lcd_y_size;

/* The following values are same for portrait and landscape orientations */
VSA = OTM8009A_480X800_VSYNC; /* 12 */
VBP = OTM8009A_480X800_VBP; /* 12 */
VFP = OTM8009A_480X800_VFP; /* 12 */
HSA = OTM8009A_480X800_HSYNC; /* 63 */
HBP = OTM8009A_480X800_HBP; /* 20 */
HFP = OTM8009A_480X800_HFP; /* 20 */

hdsivideo_handle.VirtualChannelID = LCD_OTM8009A_ID;
hdsivideo_handle.ColorCoding = LCD_DSI_PIXEL_DATA_FMT_RBG888;
hdsivideo_handle.VSPolarity = DSI_VSYNC_ACTIVE_HIGH;
hdsivideo_handle.HSPolarity = DSI_HSYNC_ACTIVE_HIGH;
hdsivideo_handle.DEPolarity = DSI_DATA_ENA
BLE_ACTIVE_HIGH;
00371  hdsivideo_handle.Mode = DSI_VID_MODE_BURST;
       /* Mode Video burst ie : one LgP per line */
00372  hdsivideo_handle.NullPacketSize = 0xFFF;
00373  hdsivideo_handle.NumberOfChunks = 0;
00374  hdsivideo_handle.PacketSize = HACT;  /* Value depending on display orientation choice portrait/landscape */
00375  hdsivideo_handle.HorizontalSyncActive = (HSA * laneByteClk_kHz)/LcdClock;
00376  hdsivideo_handle.HorizontalBackPorch = (HBP * laneByteClk_kHz)/LcdClock;
00377  hdsivideo_handle.HorizontalLine = ((HACT + HSA + HBP + HFP) * laneByteClk_kHz)/LcdClock;  /* Value depending on display orientation choice portrait/landscape */
00378  hdsivideo_handle.VerticalSyncActive = VSA;
00379  hdsivideo_handle.VerticalBackPorch = VBP;
00380  hdsivideo_handle.VerticalFrontPorch = VFP;
00381  hdsivideo_handle.VerticalActive = VACT;  /* Value depending on display orientation choice portrait/landscape */
00382
00383  /* Enable or disable sending LP command while streaming is active in video mode */
00384  hdsivideo_handle.LPCommandEnable = DSI_LP_COMMAND_ENABLE;  /* Enable sending commands in mode LP (Low Power) */
00385
00386  /* Largest packet size possible to transmit in LP mode in VSA, VBP, VFP regions */
00387  /* Only useful when sending LP packets is allowed while streaming is active in video mode */
00388  hdsivideo_handle.LPLargestPacketSize = 16;
/* Largest packet size possible to transmit in LP mode in HFP region during VACT period */
/* Only useful when sending LP packets is allowed while streaming is active in video mode */

hdsivideo_handle.LPVACTLargestPacketSize = 0;

/* Specify for each region of the video frame, if the transmission of command in LP mode is allowed in this region */
/* while streaming is active in video mode */

hdsivideo_handle.LPHorizontalFrontPorchEnable = DSI_LP_HFP_ENABLE;  /* Allow sending LP commands during HFP period */

hdsivideo_handle.LPHorizontalBackPorchEnable = DSI_LP_HBP_ENABLE;  /* Allow sending LP commands during HBP period */

hdsivideo_handle.LPVerticalActiveEnable = DSI_LP_VACT_ENABLE;  /* Allow sending LP commands during VACT period */

hdsivideo_handle.LPVerticalFrontPorchEnable = DSI_LP_VFP_ENABLE;  /* Allow sending LP commands during VFP period */

hdsivideo_handle.LPVerticalBackPorchEnable = DSI_LP_VBP_ENABLE;  /* Allow sending LP commands during VBP period */

hdsivideo_handle.LPVerticalSyncActiveEnable = DSI_LP_VSYNC_ENABLE;  /* Allow sending LP commands during VSync = VSA period */

/* Configure DSI Video mode timings with settings set above */

HAL_DSI_ConfigVideoMode(&hdsi_discovery), &hdsivideo_handle);
00405 00406 /**********************************************************************************End DSI Initialization**********************************************************************************/
00407 00408 00409 00410 00411 00412 00413 00414 00415 00416 00417 00418 00419 00420 00421 00422 00423 00424 00425 00426 00427
00409 /**************************LTDC Initialization******************************************************************************************/

00410 /* Timing Configuration */
00411 hltdc_discovery.Init.HorizontalSync = (HSA - 1);
00412 hltdc_discovery.Init.AccumulatedHBP = (HSA + HBP - 1);
00413 hltdc_discovery.Init.AccumulatedActiveW = (lcd_x_size + HSA + HBP - 1);
00414 hltdc_discovery.Init.TotalWidth = (lcd_x_size + HSA + HBP + HFP - 1);
00415
00416 /* Initialize the LCD pixel width and pixel height */
00417 hltdc_discovery.LayerCfg->ImageWidth = lcd_x_size;
00418 hltdc_discovery.LayerCfg->ImageHeight = lcd_y_size;
00419
00420 /** LCD clock configuration */
00421 * Note: The following values should not be changed as the PLLSAI is also used to clock the USB FS
00422 * PLLSAI_VCO Input = HSE_VALUE/PLL_M = 1 Mhz
00423 * PLLSAI_VCO Output = PLLSAI_VCO Input * PLLSAIN = 384 Mhz
00424 * PLLLCDCLK = PLLSAI_VCO Output/PLLSAIR = 384 MHz / 7 = 54.85 MHz
00425 * LTDC clock frequency = PLLLCDCLK / LTD_C_PLLSAI_DIVR_2 = 54.85 MHz / 2 = 27.429 MHz
PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_LTDC;
PeriphClkInitStruct.PLLSAI.PLLSAIN = 384;
PeriphClkInitStruct.PLLSAI.PLLSAIR = 7;
PeriphClkInitStruct.PLLSAIDivR = RCC_PLLSAIDIVR_2;
HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct);

/* Background value */
hltdc_discovery.Init.Backcolor.Blue = 0;
hltdc_discovery.Init.Backcolor.Green = 0;
hltdc_discovery.Init.Backcolor.Red = 0;
hltdc_discovery.Init.PCPolarity = LTDC_PCPOLARITY_IPC;
hltdc_discovery.Instance = LTDC;

/* Get LTDC Configuration from DSI Configuration */
HAL_LTDC_StructInitFromVideoConfig(&hltdc_discovery), &hdsivideo_handle);

/* Initialize the LTDC */
HAL_LTDC_Init(&hltdc_discovery);

/* Enable the DSI host and wrapper after the LTDC initialization
To avoid any synchronization issue, the DSI shall be started after enabling the LTDC */
HAL_DSI_Start(&hdsi_discovery);

#if !defined(DATA_IN_ExtSDRAM)
/* Initialize the SDRAM */
BSP_SDRAM_Init();
#endif /* DATA_IN_ExtSDRAM */

/* ... */
/* Initialize the font */
BSP_LCD_SetFont(&LCD_DEFAULT_FONT);

/****************************End LTDC Initialization***************************/

/********************OTM8009A Initialization******************************/

/* Initialize the OTM8009A LCD Display IC Driver (KoD LCD IC Driver)
 * depending on configuration set in 'hdsi_video_handle'.
 */
OTM8009A_Init(OTM8009A_FORMAT_RGB888, orientation);

/****************************End OTM8009A Initialization***************************/

return LCD_OK;
}

#if defined(USE_LCD_HDMI)
/**
 * @brief Initializes the DSI for HDMI monitor.
 * The initialization is done as below:
 * - DSI PLL initialization
 * - DSI initialization
 * - LTDC initialization
 * - DSI-HDMI ADV7533 adapter device initialization
 * @param format : HDMI format could be HDMI_FORMAT_720_480 or HDMI_FORMAT_720_576
 * @retval LCD state
uint8_t BSP_LCD_HDMIInitEx(uint8_t format) {
  /************************ADV7533 Initialization**************************/
  /*
   * Initialize the ADV7533 HDMI Bridge depending on configuration set in 'hdsi_video_handle'.
   */
  adv7533ConfigTypeDef adv7533_config;
  adv7533_config.DSI_LANES = 2;
  adv7533_config.HACT = HDMI_Format[format].HACT;
  adv7533_config.HSYNC = HDMI_Format[format].HSYNC;
  adv7533_config.HBP = HDMI_Format[format].HBP;
  adv7533_config.HFP = HDMI_Format[format].HFP;
  adv7533_config.VACT = HDMI_Format[format].VACT;
  adv7533_config.VSYNC = HDMI_Format[format].VSYNC;
  adv7533_config.VBP = HDMI_Format[format].VBP;
  adv7533_config.VFP = HDMI_Format[format].VFP;
  ADV7533_Init();
  ADV7533_Configure(&adv7533_config);
  ADV7533_PowerOn();
  /************************ Update hdmi_x_size and hdmi_y_size **************************/
  lcd_x_size = HDMI_Format[format].HACT;
lcd_y_size = HDMI_Format[format].VACT;

/********************End ADV7533 Initialization***************************/
DSI_PLLInitTypeDef dsiPllInit;
DSI_PHY_TimerTypeDef dsiPhyInit;
static RCC_PeriphCLKInitTypeDefPeriphClkInitStruct;

/* Call first MSP Initialize only in case of first initialization
 * This will set IP blocks LTDC and DSI
 * - out of reset
 * - clocked
 * - NVIC IRQ related to IP blocks enabled */
BSP_LCD_MspInit();

/****************************DSI Initialization*****************************/
/* Base address of DSI Host/Wrapper registers to be set before calling De-Init */
hdsi_discovery.Instance = DSI;
HAL_DSI_DeInit(&(hdsi_discovery));

/* Configure the DSI PLL */
dsiPllInit.PLLNDIV = HDMI_PLLConfig[format].NDIV;
dsiPllInit.PLLIDF = HDMI_PLLConfig[format].IDF;
dsiPllInit.PLLODF = HDMI_PLLConfig[format].ODF;

/* Set number of Lanes */
hdsi_discovery.Init.NumberOfLanes = DSI_TW
O_DATA_LANES;
00540  /* Set the TX escape clock division ratio */
00541  hdsi_discovery.Init.TXEscapeCkdiv = HDMI_PLLConfig[format].TXEscapeCkdiv;
00542  /* Disable the automatic clock lane control (the ADV7533 must be clocked) */
00543  hdsi_discovery.Init.AutomaticClockLaneControl = DSI_AUTO_CLK_LANE_CTRL_DISABLE;
00544
00545  /* Init the DSI */
00546  HAL_DSI_Init(&hdsi_discovery, &dsiPllInit);
00547
00548  /* Configure the D-PHY Timings */
00549  dsiPhyInit.ClockLaneHS2LPTime = 0x14;
00550  dsiPhyInit.ClockLaneLP2HSTime = 0x14;
00551  dsiPhyInit.DataLaneHS2LPTime = 0x0A;
00552  dsiPhyInit.DataLaneLP2HSTime = 0x0A;
00553  dsiPhyInit.DataLaneMaxReadTime = 0x00;
00554  dsiPhyInit.StopWaitTime = 0x0;
00555  HAL_DSI_ConfigPhyTimer(&hdsi_discovery, &dsiPhyInit);
00556
00557  /* Virutal channel used by the ADV7533 */
00558  hdsivideo_handle.VirtualChannelID = HDMI_ADV7533_ID;
00559
00560  /* Timing parameters for Video modes */
00561  /* Set Timing parameters of DSI depending on its chosen format */
00562  hdsivideo_handle.ColorCoding = HDMI_Format[format].RGB_CODING;
00563  hdsivideo_handle.LooselyPacked = DSI_LOOSELY_PACKED_DISABLE;
00564  hdsivideo_handle.VSPolarity = DSI_VSYNC_ACTIVE_LOW;
hdsivideo_handle.HSPolarity = DS I_HSYNC_ACTIVE_LOW;
hdsivideo_handle.DEPolarity = DS I_DATA_ENABLE_ACTIVE_HIGH;
hdsivideo_handle.Mode = DS I_VID_MODE_NB_PULSES;

hdsivideo_handle.NullPacketSize = HD MI_DSIPacket[format].NullPacketSize;

hdsivideo_handle.NumberOfChunks = HD MI_DSIPacket[format].NumberOfChunks;

hdsivideo_handle.PacketSize = HD MI_DSIPacket[format].PacketSize;

hdsivideo_handle.HorizontalSyncActive = HD MI_Format[format].HSYNC*HDMI_PLLConfig[format].LaneByteClock/HDMI_PLLConfig[format].PCLK;

hdsivideo_handle.HorizontalBackPorch = HD MI_Format[format].HBP*HDMI_PLLConfig[format].LaneByteClock/HDMI_PLLConfig[format].PCLK;

hdsivideo_handle.HorizontalLine = (HDMI_Format[format].HACT + HDMI_Format[format].HSYNC + HDMI_Format[format].HBP + HDMI_Format[format].HFP)*HDMI_PLLConfig[format].LaneByteClock/HDMI_PLLConfig[format].PCLK;

hdsivideo_handle.VerticalSyncActive = HD MI_Format[format].VSYNC;

hdsivideo_handle.VerticalBackPorch = HD MI_Format[format].VBP;

hdsivideo_handle.VerticalFrontPorch = HD MI_Format[format].VFP;

hdsivideo_handle.VerticalActive = HD MI_Format[format].VACT;

/* Enable or disable sending LP command while streaming is active in video mode */

hdsivideo_handle.LPCommandEnable = DS I_LP_COMMAND_DISABLE; /* Enable sending commands in mode LP (Low Power) */
/* Largest packet size possible to transmit in LP mode in VSA, VBP, VFP regions */
/* Only useful when sending LP packets is allowed while streaming is active in video mode */
hdsivideo_handle.LPLargestPacketSize = 4;

/* Largest packet size possible to transmit in LP mode in HFP region during VACT period */
/* Only useful when sending LP packets is allowed while streaming is active in video mode */
hdsivideo_handle.LPVACTLargestPacketSize = 4;

/* Specify for each region, if the going in LP mode is allowed */
/* while streaming is active in video mode */
hdsivideo_handle.LPHorizontalFrontPorchEnable = DSI_LP_HFP_DISABLE;
hdsivideo_handle.LPHorizontalBackPorchEnable = DSI_LP_HBP_DISABLE;
hdsivideo_handle.LPVerticalActiveEnable = DSI_LP_VACT_DISABLE;
hdsivideo_handle.LPVerticalFrontPorchEnable = DSI_LP_VFP_DISABLE;
hdsivideo_handle.LPVerticalBackPorchEnable = DSI_LP_VBP_DISABLE;
hdsivideo_handle.LPVerticalSyncActiveEnable = DSI_LP_VSYNC_DISABLE;

/* No acknowledge at the end of a frame */
hdsivideo_handle.FrameBTAAcknowledgeEnable = DSI_FBTAA_DISABLE;

/* Configure DSI Video mode timings with s
settings set above */
00603  HAL_DSI_ConfigVideoMode(&hdsi_discovery, &hdsivideo_handle);
00604
00605  /* Enable the DSI host and wrapper : but LTDC is not started yet at this stage */
00606  HAL_DSI_Start(&hdsi_discovery);
00607
00608  /**************************End DSI Initialization***************************/
00609 00610
00611  /**************************LTDC Initialization***************************/
00612
00613  /* LTDC clock configuration */
00614 PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_LTDC;
00615 PeriphClkInitStruct.PLLSAI.PLLSAIN = HDMI_PLLConfig[format].PLLSAIN;
00616 PeriphClkInitStruct.PLLSAI.PLLSAIR = HDMI_PLLConfig[format].PLLSAIR;
00617 PeriphClkInitStruct.PLLSAIDivR = RCC_PLLSAIDIVR_2;
00618  HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct);
00619
00620  /* Base address of LTDC registers to be set before calling De-Init */
00621  hltdc_discovery.Instance = LTDC;
00622
00623  HAL_LTDC_DeInit(&hltdc_discovery));
00624
00625  /* Timing Configuration */
00626  hltdc_discovery.Init.HorizontalSync = (HDMI_I_Format[format].HSYNC - 1);
00627  hltdc_discovery.Init.AccumulatedHBP = (HDMI
I_Format[format].HSYNC + HDMI_Format[format].HBP - 1);  
00628   hltdc_discovery.Init.AccumulatedActiveW = (HDMI_Format[format].HACT + HDMI_Format[format].HSYNC + HDMI_Format[format].HBP - 1);  
00629   hltdc_discovery.Init.TotalWidth = (HDMI_Format[format].HACT + HDMI_Format[format].HSYNC + HDMI_Format[format].HBP + HDMI_Format[format].HFP - 1);  
00630   hltdc_discovery.Init.VerticalSync = (HDMI_Format[format].VSYNC - 1);  
00631   hltdc_discovery.Init.AccumulatedVBP = (HDMI_Format[format].VSYNC + HDMI_Format[format].VBP + HDMI_Format[format].HFP - 1);  
00632   hltdc_discovery.Init.AccumulatedActiveH = (HDMI_Format[format].VACT + HDMI_Format[format].VSYNC + HDMI_Format[format].VBP - 1);  
00633   hltdc_discovery.Init.TotalHeight = (HDMI_Format[format].VACT + HDMI_Format[format].VSYNC + HDMI_Format[format].VBP + HDMI_Format[format].VFP - 1);  
00634   hltdc_discovery.Init.Backcolor.Blue = 0x00;  
00635   hltdc_discovery.Init.Backcolor.Green = 0xFF;  
00636   hltdc_discovery.Init.Backcolor.Red = 0xFF;  
00637   hltdc_discovery.Init.HSPolarity = LTDC_HSPOLARITY_AL;  
00638   hltdc_discovery.Init.VSPolarity = LTDC_VSPOLARITY_AL;  
00639   hltdc_discovery.Init.DEPolarity = LTDC_DEPOLARITY_AL;  
00640   hltdc_discovery.Init.PCPolarity = LTDC_PCP
/* Initialize & Start the LTDC */
HAL_LTDC_Init(&hltdc_discovery);

#if !defined(DATA_IN_ExtSDRAM)
/* Initialize the SDRAM */
BSP_SDRAM_Init();
#endif /* DATA_IN_ExtSDRAM */

/* Initialize the font */
BSP_LCD_SetFont(&LCD_DEFAULT_FONT);

/************************End LTDC Initialization******************************/

return LCD_OK;
#endif /* USE_LCD_HDMI */

/**
  * @brief  BSP LCD Reset
  * Hw reset the LCD DSI activating its XRES signal (active low for some time)
  * and desactivating it later.
  */

void BSP_LCD_Reset(void)
{
    GPIO_InitTypeDef gpio_init_structure;

    __HAL_RCC_GPIOJ_CLK_ENABLE();

    /* Configure the GPIO on PJ15 */
    gpio_init_structure.Pin = GPIO_PIN_15;
    gpio_init_structure.Mode = GPIO_MODE_OUTPUT_PP;
    gpio_init_structure.Pull = GPIO_PULLUP;
    gpio_init_structure.Speed = GPIO_SPEED_H...
IGH;

00678 HAL_GPIO_Init(GPIOJ, &gpio_init_structure);
00680
00681 /* Activate XRES active low */
00682 HAL_GPIO_WritePin(GPIOJ, GPIO_PIN_15, GPIO_PIN_RESET);
00683
00684HAL_Delay(20); /* wait 20 ms */
00685
00686 /* Desactivate XRES */
00687 HAL_GPIO_WritePin(GPIOJ, GPIO_PIN_15, GPIO_PIN_SET);
00688
00689 /* Wait for 10ms after releasing XRES before sending commands */
00690HAL_Delay(10);
00691}
00692
00693 /**
00694 * @brief Gets the LCD X size.
00695 * @retval Used LCD X size
00696 */
00697 uint32_t BSP_LCD_GetXSize(void)
00698 {
00699 return (lcd_x_size);
00700}
00701
00702 /**
00703 * @brief Gets the LCD Y size.
00704 * @retval Used LCD Y size
00705 */
00706 uint32_t BSP_LCD_GetYSize(void)
00707 {
00708 return (lcd_y_size);
00709}
void BSP_LCD_SetXSize(uint32_t imageWidthPixels)
{
    hltdc_discovery.LayerCfg[ActiveLayer].ImageWidth = imageWidthPixels;
}

void BSP_LCD_SetYSize(uint32_t imageHeightPixels)
{
    hltdc_discovery.LayerCfg[ActiveLayer].ImageHeight = imageHeightPixels;
}

void BSP_LCD_LayerDefaultInit(uint16_t LayerIndex, uint32_t FB_Address)
{

LCD_LayerCfgTypeDef Layercfg;

/* Layer Init */
Layercfg.WindowX0 = 0;
Layercfg.WindowX1 = BSP_LCD_GetXSize();
Layercfg.WindowY0 = 0;
Layercfg.WindowY1 = BSP_LCD_GetYSize();
Layercfg.PixelFormat = LTDC_PIXEL_FORMAT_RGB8888;
Layercfg.FBStartAdress = FB_Address;
Layercfg.Alpha = 255;
Layercfg.Alpha0 = 0;
Layercfg.Backcolor.Blue = 0;
Layercfg.Backcolor.Green = 0;
Layercfg.Backcolor.Red = 0;
Layercfg.BlendingFactor1 = LTDC_BLENDING_FACTOR1_PAxCA;
Layercfg.BlendingFactor2 = LTDC_BLENDING_FACTOR2_PAxCA;
Layercfg.ImageWidth = BSP_LCD_GetXSize();
Layercfg.ImageHeight = BSP_LCD_GetYSize();
HAL_LTDC_ConfigLayer(&hltdc_discovery, &Layercfg, LayerIndex);

DrawProp[LayerIndex].BackColor = LCD_COLOR_WHITE;
DrawProp[LayerIndex].pFont = &Font24;
DrawProp[LayerIndex].TextColor = LCD_COLOR_BLACK;
}

/**
 * @brief Selects the LCD Layer.
 * @param LayerIndex: Layer foreground or background

void BSP_LCD_SelectLayer(uint32_t LayerIndex) {
    ActiveLayer = LayerIndex;
}

/**
 * @brief  Sets an LCD Layer visible
 * @param  LayerIndex: Visible Layer
 * @param  State: New state of the specified layer
 *          This parameter can be one of the following values:
 *          @arg  ENABLE
 *          @arg  DISABLE
 */
void BSP_LCD_SetLayerVisible(uint32_t LayerIndex, FunctionalState State) {
    if (State == ENABLE) {
        __HAL_LTDC_LAYER_ENABLE((&hltdc_discovery), LayerIndex);
    } else {
        __HAL_LTDC_LAYER_DISABLE((&hltdc_discovery), LayerIndex);
    }
    __HAL_LTDC_RELOAD_CONFIG((&hltdc_discovery));
}

* @brief  Configures the transparency.
```c
void BSP_LCD_SetTransparency(uint32_t LayerIndex, uint8_t Transparency)
{
    HAL_LTDC_SetAlpha(&hltdc_discovery, Transparency, LayerIndex);
}

void BSP_LCD_SetLayerAddress(uint32_t LayerIndex, uint32_t Address)
{
    HAL_LTDC_SetAddress(&hltdc_discovery, Address, LayerIndex);
}

void BSP_LCD_SetDisplayWindow(uint32_t LayerIndex, uint32_t Xpos, uint32_t Ypos)
{
    HAL_LTDC_SetWindow(&hltdc_discovery, Xpos, Ypos, LayerIndex);
}
```
```c
void BSP_LCD_SetLayerWindow(uint16_t LayerIndex, uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) {
    /* Reconfigure the layer size */
    HAL_LTDC_SetWindowSize(&(hltdc_discovery), Width, Height, LayerIndex);

    /* Reconfigure the layer position */
    HAL_LTDC_SetWindowPosition(&(hltdc_discovery), Xpos, Ypos, LayerIndex);
}

void BSP_LCD_SetColorKeying(uint32_t LayerIndex, uint32_t RGBValue) {
    /* Configure and Enable the color Keying for LCD Layer */
    HAL_LTDC_ConfigColorKeying(&(hltdc_discovery), RGBValue, LayerIndex);
    HAL_LTDC_EnableColorKeying(&(hltdc_discovery), LayerIndex);
}
```

* @param Width: LCD window width
* @param Height: LCD window height
* @param LayerIndex: Layer foreground or background
* @param RGBValue: Color reference

* @brief Configures and sets the color keying.
* @brief Disables the color keying.
void BSP_LCD_ResetColorKeying(uint32_t LayerIndex) {
  /* Disable the color Keying for LCD Layer */
  HAL_LTDC_DisableColorKeying(&hltdc_discovery, LayerIndex);
}

void BSP_LCD_SetTextColor(uint32_t Color) {
  DrawProp[ActiveLayer].TextColor = Color;
}

uint32_t BSP_LCD_GetTextColor(void) {
  return DrawProp[ActiveLayer].TextColor;
}

void BSP_LCD_SetBackColor(uint32_t Color) { /* @brief Sets the LCD background color. */
  /* @param Color: Layer background color code ARGB(8-8-8-8) */
}

uint32_t BSP_LCD_GetBackColor(void) { /* @brief Gets the LCD background color. */
  /* @param Color: Layer background color code ARGB(8-8-8-8) */
}

* @param LayerIndex: Layer foreground or background */

* @brief Sets the LCD text color. */
* @param Color: Text color code ARGB(8-8-8-8) */

* @brief Gets the LCD text color. */
* @retval Used text color. */

* @brief Sets the LCD background color. */
* @param Color: Layer background color code ARGB(8-8-8-8) */
DrawProp[ActiveLayer].BackColor = Color;

/**
 * @brief Gets the LCD background color.
 * @retval Used background color
 */
uint32_t BSP_LCD_GetBackColor(void) {
    return DrawProp[ActiveLayer].BackColor;
}

/**
 * @brief Sets the LCD text font.
 * @param fonts: Layer font to be used
 */
void BSP_LCD_SetFont(sFONT *fonts) {
    DrawProp[ActiveLayer].pFont = fonts;
}

/**
 * @brief Gets the LCD text font.
 * @retval Used layer font
 */
sFONT *BSP_LCD_GetFont(void) {
    return DrawProp[ActiveLayer].pFont;
}

/**
 * @brief Reads an LCD pixel.
 * @param Xpos: X position
 * @param Ypos: Y position
 * @retval RGB pixel color
 */
uint32_t BSP_LCD_ReadPixel(uint16_t Xpos, uint16_t Ypos)
{
    uint32_t ret = 0;
    if(hltdc_discovery.LayerCfg[ActiveLayer].PixelFormat == LTDC_PIXEL_FORMAT_ARGB8888)
    {
        /* Read data value from SDRAM memory */
        ret = *(__IO uint32_t*) (hltdc_discovery.LayerCfg[ActiveLayer].FBStartAdress + (4*(Ypos*BSP_LCD_GetXSize()) + Xpos));
    }
    else if(hltdc_discovery.LayerCfg[ActiveLayer].PixelFormat == LTDC_PIXEL_FORMAT_RGB888)
    {
        /* Read data value from SDRAM memory */
        ret = (*(__IO uint32_t*) (hltdc_discovery.LayerCfg[ActiveLayer].FBStartAdress + (4*(Ypos*BSP_LCD_GetXSize()) + Xpos))) & 0x00FFFFFF;
    }
    else if((hltdc_discovery.LayerCfg[ActiveLayer].PixelFormat == LTDC_PIXEL_FORMAT_RGB565) ||
             (hltdc_discovery.LayerCfg[ActiveLayer].PixelFormat == LTDC_PIXEL_FORMAT_ARGB4444) ||
             (hltdc_discovery.LayerCfg[ActiveLayer].PixelFormat == LTDC_PIXEL_FORMAT_AL88))
    {
        /* Read data value from SDRAM memory */
        ret = *__IO uint16_t* (hltdc_discovery.LayerCfg[ActiveLayer].FBStartAdress + (2*(Ypos*BSP_LCD_GetXSize()) + Xpos));
    }
    else
    {
        /* Read data value from SDRAM memory */
    }
```c
ret = *(__IO uint8_t*) (hltdc_discovery.LayerCfg[ActiveLayer].FBStartAdress + (2*(Ypos*BSP_LCD_GetXSize() + Xpos)));
}

return ret;
}

void BSP_LCD_Clear(uint32_t Color)
{
    /* Clear the LCD */
    LL_FillBuffer(ActiveLayer, (uint32_t *)(hltdc_discovery.LayerCfg[ActiveLayer].FBStartAdress), BSP_LCD_GetXSize(), BSP_LCD_GetYSize(), 0, Color);
}

void BSP_LCD_ClearStringLine(uint32_t Line)
{
    uint32_t color_backup = DrawProp[ActiveLayer].TextColor;
    DrawProp[ActiveLayer].TextColor = DrawProp[ActiveLayer].BackColor;

    /* Draw rectangle with background color */
    BSP_LCD_FillRect(0, (Line * DrawProp[ActiveLayer].pFont->Height), BSP_LCD_GetXSize(), DrawPr
op[ActiveLayer].pFont->Height);

DrawProp[ActiveLayer].TextColor = color_backup;
BSP_LCD_SetTextColor(DrawProp[ActiveLayer].TextColor);
}

/** *
 * @brief Displays one character in currently active layer.
 * @param Xpos: Start column address
 * @param Ypos: Line where to display the character shape.
 * @param Ascii: Character ascii code
 * This parameter must be a number between Min_Data = 0x20 and Max_Data = 0x7E
 */
void BSP_LCD_DisplayChar(uint16_t Xpos, uint16_t Ypos, uint8_t Ascii)
{
    DrawChar(Xpos, Ypos, &DrawProp[ActiveLayer].pFont->table[(Ascii-' ')] *\
        DrawProp[ActiveLayer].pFont->Height * ((DrawProp[ActiveLayer].pFont->Width + 7) / 8));
}

/** *
 * @brief Displays characters in currently active layer.
 * @param Xpos: X position (in pixel)
 * @param Ypos: Y position (in pixel)
 * @param Text: Pointer to string to display on LCD
 * @param Mode: Display mode
 * This parameter can be one of the following values:
 *
void BSP_LCD_DisplayStringAt(uint16_t Xpos, uint16_t Ypos, uint8_t *Text, Text_AlignModeTypdef Mode)
{
    uint16_t refcolumn = 1, i = 0;
    uint32_t size = 0, xsize = 0;
    uint8_t *ptr = Text;

    /* Get the text size */
    while (*ptr++) size ++ ;

    /* Characters number per line */
    xsize = (BSP_LCD_GetXSize()/DrawProp[ActiveLayer].pFont->Width);

    switch (Mode) {
    case CENTER_MODE:
        refcolumn = Xpos + ((xsize - size)*DrawProp[ActiveLayer].pFont->Width) / 2;
        break;
    case LEFT_MODE:
        refcolumn = Xpos;
        break;
    case RIGHT_MODE:
        refcolumn = - Xpos + ((xsize - size)*DrawProp[ActiveLayer].pFont->Width);
        break;
    }
default:
{
    refcolumn = Xpos;
    break;
}

/* Check that the Start column is located in the screen */
if ((refcolumn < 1) || (refcolumn >= 0x800))
{
    refcolumn = 1;
}

/* Send the string character by character on LCD */
while (*Text != 0) & ((BSP_LCD_GetXSize() - (i*DrawProp[ActiveLayer].pFont->Width)) & 0xFF FF) >= DrawProp[ActiveLayer].pFont->Width)
{
    /* Display one character on LCD */
    BSP_LCD_DisplayChar(refcolumn, Ypos, *Text);
    /* Decrement the column position by 16 */
    refcolumn += DrawProp[ActiveLayer].pFont->Width;
}

/* Point on the next character */
Text++; i++;
/**
   * @brief Displays a maximum of 60 characters on the LCD.
   * @param Line: Line where to display the character shape
   * @param ptr: Pointer to string to display on LCD
   */
void BSP_LCD_DisplayStringAtLine(uint16_t Line, uint8_t *ptr)
{
    BSP_LCD_DisplayStringAt(0, LINE(Line), ptr, LEFT_MODE);
}

/**
   * @brief Draws a horizontal line in currently active layer.
   * @param Xpos: X position
   * @param Ypos: Y position
   * @param Length: Line length
   */
void BSP_LCD_DrawHLine(uint16_t Xpos, uint16_t Ypos, uint16_t Length)
{
    uint32_t Xaddress = 0;
    /* Get the line address */
    Xaddress = (hltdc_discovery.LayerCfg[ActiveLayer].FBStartAdress) + 4*(BSP_LCD_GetXSize() * Ypos + Xpos);
    /* Write line */
    LL_FillBuffer(ActiveLayer, (uint32_t *)Xaddress, Length, 1, 0, DrawProp[ActiveLayer].TextColor);
}
/**
 * @brief Draws a vertical line in currently active layer.
 * @param Xpos: X position
 * @param Ypos: Y position
 * @param Length: Line length
 */

void BSP_LCD_DrawVLine(uint16_t Xpos, uint16_t Ypos, uint16_t Length)
{
    uint32_t Xaddress = 0;

    /* Get the line address */
    Xaddress = (hltdc_discovery.LayerCfg[ActiveLayer].FBStartAdress) + 4*(BSP_LCD_GetXSize() * Ypos + Xpos);

    /* Write line */
    LL_FillBuffer(ActiveLayer, (uint32_t *)Xaddress, 1, Length, (BSP_LCD_GetXSize() - 1), DrawProp[ActiveLayer].TextColor);
}

/**
 * @brief Draws an uni-line (between two points) in currently active layer.
 * @param x1: Point 1 X position
 * @param y1: Point 1 Y position
 * @param x2: Point 2 X position
 * @param y2: Point 2 Y position
 */

void BSP_LCD_DrawLine(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2)
{  
    int16_t deltax = 0, deltay = 0, x = 0, y = 0, xinc1 = 0, xinc2 = 0,
yinc1 = 0, yinc2 = 0, den = 0, num = 0, numadd = 0, numpixels = 0,
curpixel = 0;
deltax = \text{ABS}(x_2 - x_1); /* The difference between the x's */
deltay = \text{ABS}(y_2 - y_1); /* The difference between the y's */
x = x_1; /* Start x off at the first pixel */
y = y_1; /* Start y off at the first pixel */
if (x_2 \geq x_1) /* The x-values are increasing */
{
xinc1 = 1;
xinc2 = 1;
}
else /* The x-values are decreasing */
{
xinc1 = -1;
xinc2 = -1;
}
if (y_2 \geq y_1) /* The y-values are increasing */
{
yinc1 = 1;
yinc2 = 1;
}
else /* The y-values are decreasing */
{
yinc1 = -1;
yinc2 = -1;
if (deltax >= deltay) /* There is at least one x-value for every y-value */
{
    xinc1 = 0; /* Don't change the x when numerator >= denominator */
    yinc2 = 0; /* Don't change the y for every iteration */
    den = deltay;
    num = deltay / 2;
    numadd = deltay;
    numpixels = deltay; /* There are more x-values than y-values */
}
else /* There is at least one y-value for every x-value */
{
    xinc2 = 0; /* Don't change the x for every iteration */
    yinc1 = 0; /* Don't change the y when numerator >= denominator */
    den = deltay;
    num = deltay / 2;
    numadd = deltay;
    numpixels = deltay; /* There are more y-values than x-values */
}
for (curpixel = 0; curpixel <= numpixels; curpixel++)
{
    BSP_LCD_DrawPixel(x, y, DrawProp[ActiveLayer].TextColor); /* Draw the current pixel */
    num += numadd; /* Increase the numerator by the top of the fraction */
if (num >= den)
/* Check if numerator >= denominator */
{
    num -= den;
    /* Calculate the new numerator value */
    x += xinc1;
    /* Change the x as appropriate */
    y += yinc1;
    /* Change the y as appropriate */
}

x += xinc2;
/* Change the x as appropriate */
y += yinc2;
/* Change the y as appropriate */

/**
 * @brief Draws a rectangle in currently active layer.
 * @param Xpos: X position
 * @param Ypos: Y position
 * @param Width: Rectangle width
 * @param Height: Rectangle height
 */
void BSP_LCD_DrawRect(uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height)
{
    /* Draw horizontal lines */
    BSP_LCD_DrawHLine(Xpos, Ypos, Width);
    BSP_LCD_DrawHLine(Xpos, (Ypos + Height), Width);

    /* Draw vertical lines */
    BSP_LCD_DrawVLine(Xpos, Ypos, Height);
    BSP_LCD_DrawVLine((Xpos + Width), Ypos, Height);
/**
 * @brief Draws a circle in currently active layer.
 * @param Xpos: X position
 * @param Ypos: Y position
 * @param Radius: Circle radius
 */

void BSP_LCD_DrawCircle(uint16_t Xpos, uint16_t Ypos, uint16_t Radius) {
    int32_t D;    /* Decision Variable */
    uint32_t CurX; /* Current X Value */
    uint32_t CurY; /* Current Y Value */
    D = 3 - (Radius << 1);
    CurX = 0;
    CurY = Radius;
    while (CurX <= CurY) {
        BSP_LCD_DrawPixel((Xpos + CurX), (Ypos - CurY), DrawProp[ActiveLayer].TextColor);
        BSP_LCD_DrawPixel((Xpos - CurX), (Ypos - CurY), DrawProp[ActiveLayer].TextColor);
        BSP_LCD_DrawPixel((Xpos + CurY), (Ypos - CurX), DrawProp[ActiveLayer].TextColor);
        BSP_LCD_DrawPixel((Xpos - CurY), (Ypos - CurX), DrawProp[ActiveLayer].TextColor);
        BSP_LCD_DrawPixel((Xpos + CurX), (Ypos + CurY), DrawProp[ActiveLayer].TextColor);
        CurX++;
    }
}
BSP_LCD_DrawPixel((Xpos - CurX), (Ypos + CurY), DrawProp[ActiveLayer].TextColor);
BSP_LCD_DrawPixel((Xpos + CurY), (Ypos + CurX), DrawProp[ActiveLayer].TextColor);
BSP_LCD_DrawPixel((Xpos - CurY), (Ypos + CurX), DrawProp[ActiveLayer].TextColor);

if (D < 0)
{
    D += (CurX << 2) + 6;
}
else
{
    D += ((CurX - CurY) << 2) + 10;
    CurY--;
}
CurX++;
}

/**
 * @brief Draws an poly-line (between many points) in currently active layer.
 * @param Points: Pointer to the points array
 * @param PointCount: Number of points
 */
void BSP_LCD_DrawPolygon(pPoint Points, uint16_t PointCount)
{
    int16_t X = 0, Y = 0;
    if(PointCount < 2)
    {
        return;
    }
BSP_LCD_DrawLine(Points->X, Points->Y, (Points+PointCount-1)->X, (Points+PointCount-1)->Y);
while(--PointCount) {
    X = Points->X;
    Y = Points->Y;
    Points++;
    BSP_LCD_DrawLine(X, Y, Points->X, Points->Y);
}
/**
 * @brief Draws an ellipse on LCD in currently active layer.
 * @param Xpos: X position
 * @param Ypos: Y position
 * @param XRADIUS: Ellipse X radius
 * @param YRadius: Ellipse Y radius
 */
void BSP_LCD_DrawEllipse(int Xpos, int Ypos, int XRADIUS, int YRadius) {
    int x = 0, y = -YRadius, err = 2-2*XRADIUS, e2;
    float K = 0, rad1 = 0, rad2 = 0;
    rad1 = XRADIUS;
    rad2 = YRadius;
    K = (float)(rad2/rad1);
    do {
        BSP_LCD_DrawPixel((Xpos-(uint16_t)(x/K))}
01286   BSP_LCD_DrawPixel((Xpos+(uint16_t)(x/K))
01287   , (Ypos+y), DrawProp[ActiveLayer].TextColor);
01288   BSP_LCD_DrawPixel((Xpos+(uint16_t)(x/K))
01289   , (Ypos-y), DrawProp[ActiveLayer].TextColor);
01290   BSP_LCD_DrawPixel((Xpos-(uint16_t)(x/K))
01291   , (Ypos-y), DrawProp[ActiveLayer].TextColor);

01292   e2 = err;
01293   if (e2 <= x) {
01294       err += ++x*2+1;
01295       if (-y == x && e2 <= y) e2 = 0;
01296   }
01297   if (e2 > y) err += ++y*2+1;
01298   while (y <= 0);
01299
01300 /**
01301     * @brief Draws a bitmap picture loaded in the internal Flash (32 bpp) in currently active layer.
01302     * @param Xpos: Bmp X position in the LCD
01303     * @param Ypos: Bmp Y position in the LCD
01304     * @param pbmp: Pointer to Bmp picture address in the internal Flash
01305     */
01306 void BSP_LCD_DrawBitmap(uint32_t Xpos, uint32_t Ypos, uint8_t *pbmp)
01307 {
01308     uint32_t index = 0, width = 0, height = 0,
01309     bit_pixel = 0;
01310     uint32_t Address;
01311     uint32_t InputColorMode = 0;
01312     /* Get bitmap data address offset */
01313     index = *(__IO uint16_t *)(pbmp + 10);
index | = (*(__IO uint16_t *) (pbmp + 12)) << 16;

/* Read bitmap width */
width = *((uint16_t *) (pbmp + 18));
width | = (*((uint16_t *) (pbmp + 20)) << 16);

/* Read bitmap height */
height = *((uint16_t *) (pbmp + 22));
height | = (*((uint16_t *) (pbmp + 24)) << 16);

/* Read bit/pixel */
bit_pixel = *((uint16_t *) (pbmp + 28));

/* Set the address */
Address = hltdc_discovery.LayerCfg[ActiveLayer].FBStartAdress + (((BSP_LCD_GetXSize()*Ypos) + Xpos)*(4));

/* Get the layer pixel format */
if ((bit_pixel/8) == 4)
    { InputColorMode = DMA2D_INPUT_ARGB8888; }
else if ((bit_pixel/8) == 2)
    { InputColorMode = DMA2D_INPUT_RGB565; }
else
    { InputColorMode = DMA2D_INPUT_RGB888; }

/* Bypass the bitmap header */
pbmp += (index + (width * (height - 1)) * (256 - bit_pixel));
/* Convert picture to ARGB8888 pixel format */
for(index=0; index < height; index++)
{
    /* Pixel format conversion */
    LL_ConvertLineToARGB8888((uint32_t *)pbmp, (uint32_t *)Address, width, InputColorMode);

    /* Increment the source and destination buffers */
    Address += (BSP_LCD_GetXSize() * 4);
    pbmp -= width * (bit_pixel / 8);
}

/**
 * @brief Draws a full rectangle in currently active layer.
 * @param Xpos: X position
 * @param Ypos: Y position
 * @param Width: Rectangle width
 * @param Height: Rectangle height
 */
void BSP_LCD_FillRect(uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height)
{
    uint32_t Xaddress = 0;
    
    /* Set the text color */
    BSP_LCD_SetTextColor(DrawProp[ActiveLayer].TextColor);

    /* Get the rectangle start address */
    Xaddress = (hltdc_discovery.LayerCfg[ActiveLayer].FBStartAdress) + 4 * (BSP_LCD_GetXSize() * Ypos);
01375  /* Fill the rectangle */
01376  LL_FillBuffer(ActiveLayer, (uint32_t *)Xaddress, Width, Height, (BSP_LCD_GetXSize() - Width), DrawProp[ActiveLayer].TextColor);
01377 }
01378
01379 /**
01380  * @brief Draws a full circle in currently active layer.
01381  * @param Xpos: X position
01382  * @param Ypos: Y position
01383  * @param Radius: Circle radius
01384 */
01385 void BSP_LCD_FillCircle(uint16_t Xpos, uint16_t Ypos, uint16_t Radius)
01386 {
01387     int32_t D;  /* Decision Variable */
01388     uint32_t CurX; /* Current X Value */
01389     uint32_t CurY; /* Current Y Value */
01390
01391     D = 3 - (Radius << 1);
01392
01393     CurX = 0;
01394     CurY = Radius;
01395
01396     BSP_LCD_SetTextColor(DrawProp[ActiveLayer].TextColor);
01397
01398     while (CurX <= CurY)
01399     {
01400         if(CurY > 0)
01401             {
01402                 BSP_LCD_DrawHLine(Xpos - CurY, Ypos + CurX, 2*CurY);
01403                 BSP_LCD_DrawHLine(Xpos - CurY, Ypos - CurX, 2*CurY);
01404             }
01405             else
01406                 {
01407                     BSP_LCD_DrawHLine(Xpos + CurY, Ypos + CurX, 2*CurY);
01408                     BSP_LCD_DrawHLine(Xpos + CurY, Ypos - CurX, 2*CurY);
01409                 }
01410             CurX++;
01411         
01412     }
if (CurX > 0) {
    BSP_LCD_DrawHLine(Xpos - CurX, Ypos - CurY, 2*CurX);
    BSP_LCD_DrawHLine(Xpos - CurX, Ypos + CurY, 2*CurX);
}

if (D < 0) {
    D += (CurX << 2) + 6;
}
else {
    D += ((CurX - CurY) << 2) + 10;
    CurY--;  
    CurX++;
}

BSP_LCD_SetTextColor(DrawProp[ActiveLayer].TextColor);
BSP_LCD_DrawCircle(Xpos, Ypos, Radius);
}

/**
 * @brief Draws a full poly-line (between many points) in currently active layer.
 * @param Points: Pointer to the points array
 * @param PointCount: Number of points
 */
void BSP_LCD_FillPolygon(pPoint Points, uint16_t PointCount) {

int16_t X = 0, Y = 0, X2 = 0, Y2 = 0, X_center = 0, Y_center = 0, X_first = 0, Y_first = 0, pixelX = 0, pixelY = 0, counter = 0;

uint16_t IMAGE_LEFT = 0, IMAGE_RIGHT = 0, IMAGE_TOP = 0, IMAGE_BOTTOM = 0;

IMAGE_LEFT = IMAGE_RIGHT = Points->X;
IMAGE_TOP= IMAGE_BOTTOM = Points->Y;

for(counter = 1; counter < PointCount; counter++)
{
    pixelX = POLY_X(counter);
    if(pixelX < IMAGE_LEFT)
    {
        IMAGE_LEFT = pixelX;
    }
    if(pixelX > IMAGE_RIGHT)
    {
        IMAGE_RIGHT = pixelX;
    }
    pixelY = POLY_Y(counter);
    if(pixelY < IMAGE_TOP)
    {
        IMAGE_TOP = pixelY;
    }
    if(pixelY > IMAGE_BOTTOM)
    {
        IMAGE_BOTTOM = pixelY;
    }
}

if(PointCount < 2)
{
    return;
}
X_center = (IMAGE_LEFT + IMAGE_RIGHT)/2;
Y_center = (IMAGE_BOTTOM + IMAGE_TOP)/2;

X_first = Points->X;
Y_first = Points->Y;

while(--PointCount) {
  X = Points->X;
  Y = Points->Y;
  Points++;
  X2 = Points->X;
  Y2 = Points->Y;
  FillTriangle(X, X2, X_center, Y, Y2, Y_center);
  FillTriangle(X, X_center, X2, Y, Y_center, Y2);
  FillTriangle(X_center, X2, X, Y_center, Y2, Y);
}

FillTriangle(X_first, X2, X_center, Y_first, Y2, Y_center);
FillTriangle(X_first, X_center, X2, Y_first, Y_center, Y2);
FillTriangle(X_center, X2, X_first, Y_center, Y2, Y_first);

/**
 * @brief Draws a full ellipse in currently active layer.
 * @param Xpos: X position
 * @param Ypos: Y position
 * @param XRadius: Ellipse X radius

```c
void BSP_LCD_FillEllipse(int Xpos, int Ypos, int XRadius, int YRadius)
{
    int x = 0, y = -YRadius, err = 2-2*XRadius, e2;
    float K = 0, rad1 = 0, rad2 = 0;
    rad1 = XRadius;
    rad2 = YRadius;
    K = (float)(rad2/rad1);
    do
    {
        BSP_LCD_DrawHLine((Xpos-(uint16_t)(x/K)), (Ypos+y), (2*(uint16_t)(x/K) + 1));
        BSP_LCD_DrawHLine((Xpos-(uint16_t)(x/K)), (Ypos-y), (2*(uint16_t)(x/K) + 1));
        e2 = err;
        if (e2 <= x)
        {
            err += ++x*2+1;
            if (-y == x && e2 <= y) e2 = 0;
        }
        if (e2 > y) err += ++y*2+1;
    } while (y <= 0);
}
```

```
/* @param YRadius: Ellipse Y radius */
```

```
/**
 * @brief Switch back on the display if was switched off by previous call of BSP_LCD_DisplayOff().
 * Exit DSI ULPM mode if was allowe
```
and configured in Dsi Configuration.

```c
01530 void BSP_LCD_DisplayOn(void)
01531 {
01532 #if defined(USE_LCD_HDMI)
01533   if(ADV7533_ID == adv7533_drv.ReadID(ADV7533_CEC_DSI_I2C_ADDR))
01534     {
01535       return; /* Not supported for HDMI display */
01536     }
01537 else
01538   #endif /* USE_LCD_HDMI */
01539   {
01540     /* Send Display on DCS command to display */
01541     HAL_DSI_ShortWrite(&hsi_discovery),
01542       hdsivideo_handle.VirtualChannelID,
01543       DSI_DCS_SHORT_PKT_WRITE_P1,
01544       OTM8009A_CMD_DISPON,
01545       0x00);
01546   }
01547 }
01548
01549 */
01550 * @brief Switch Off the display.
01551 *            Enter DSI ULPM mode if was allowed and configured in Dsi Configuration.
01552 */
01553 void BSP_LCD_DisplayOff(void)
01554 {
01555 #if defined(USE_LCD_HDMI)
01556   if(ADV7533_ID == adv7533_drv.ReadID(ADV7533_CEC_DSI_I2C_ADDR))
01557     {
```
return; /* Not supported for HDMI display */
#endif /* USE_LCD_HDMI */
{
    /* Send Display off DCS Command to display */
    HAL_DSI_ShortWrite(&hdsi_discovery, hdsivideo_handle.VirtualChannelID, DSI_DCS_SHORT_PKT_WRITE_P1, OTM8009A_CMD_DISPOFF, 0x00);
}
#endif /* USE_LCD_HDMI */

/**
 * @brief Set the brightness value
 * @param BrightnessValue: [00: Min (black), 100 Max]
 */
void BSP_LCD_SetBrightness(uint8_t BrightnessValue)
{
#if defined(USE_LCD_HDMI)
    if(ADV7533_ID == adv7533_drv.ReadID(ADV7533_CEC_DSI_I2C_ADDR))
    {
        return; /* Not supported for HDMI display */
    }
#else
    /* Send Display on DCS command to display */
#else
    /* USE_LCD_HDMI */
#endif /* USE_LCD_HDMI */

```c
y */
01587    HAL_DSI_ShortWrite(&hdsi_discovery,
01588          LCD_OTM8009A_ID,
01589          DSI_DCS_SHORT_PKT_WRITE_P1,
01590          OTM8009A_CMD_WRDISBV,
01591          (uint16_t)(BrightnessValue * 255)/100);
01592    }
01593
01594 /**
01595    * @brief  DCS or Generic short/long write command
01596    * @param  NbrParams: Number of parameters. It indicates the write command mode:
01597    *              If inferior to 2, a long write command is performed else short.
01598    * @param  pParams: Pointer to parameter values table.
01599    * @retval HAL status
01600 */
01601 void DSI_IO_WriteCmd(uint32_t NbrParams, uint8_t *pParams)
01602 {
01603    if(NbrParams <= 1)
01604    {
01605        HAL_DSI_ShortWrite(&hdsi_discovery, LCD_OTM8009A_ID, DSI_DCS_SHORT_PKT_WRITE_P1, pParams[0], pParams[1]);
01606    }
01607    else
01608    {
01609        HAL_DSI_LongWrite(&hdsi_discovery, LCD_OTM8009A_ID, DSI_DCS_LONG_PKT_WRITE, NbrParams, pParams[NbrParams], pParams);
01610    }
01611 }```
static uint16_t LCD_IO_GetID(void)
{
    #if defined(USE_LCD_HDMI)
        HDMI_IO_Init();
        HDMI_IO_Delay(60);
        if(ADV7533_ID == adv7533_drv.ReadID(ADV7533_CEC_DSI_I2C_ADDR))
            return ADV7533_ID;
        else if(HDMI_IO_Read(LCD_DSI_ADDRESS, LCD_DSI_ID_REG) == LCD_DSI_ID)
            return LCD_DSI_ID;
        else
            return 0;
    #else
        return LCD_DSI_ID;
    #endif /* USE_LCD_HDMI */
}
**Routines**

01644  ********************************************
01645  ********************************************
01646  /**
01647  * @brief  De-Initializes the BSP LCDMsp
01648  * Application can surcharge if needed this
01649  * function implementation.
01650  */
01651  __weak void BSP_LCD_MspDeInit(void)
01652  {
01653  /**
01654  * @brief Disable IRQ of LTDC IP */
01655  HAL_NVIC_DisableIRQ(LTDC_IRQHandler);
01656  /**
01657  * @brief Disable IRQ of DMA2D IP */
01658  HAL_NVIC_DisableIRQ(DMA2D_IRQHandler);
01659  /**
01660  * @brief Disable IRQ of DSI IP */
01661  HAL_NVIC_DisableIRQ(DSI_IRQHandler);
01662  /**
01663  * @brief Force and let in reset state LTDC,
01664  * DMA2D and DSI Host + Wrapper IPs */
01665  __HAL_RCC_LTDC_FORCE_RESET();
01666  __HAL_RCC_DMA2D_FORCE_RESET();
01667  __HAL_RCC_DSI_FORCE_RESET();
01668  /**
01669  * @brief Disable the LTDC, DMA2D and DSI
01670  * Host and Wrapper clocks */
01671  __HAL_RCC_LTDC_CLK_DISABLE();
01672  __HAL_RCC_DMA2D_CLK_DISABLE();
01673  __HAL_RCC_DSI_CLK_DISABLE();
01674  }
01675  /**
01676  * @brief Initialize the BSP LCD Msp.
01677  * Application can surcharge if needed this
01678  * function implementation
01679  */
__weak void BSP_LCD_MspInit(void) {
    /** @brief Enable the LTDC clock */
    __HAL_RCC_LTDC_CLK_ENABLE();

    /** @brief Toggle Sw reset of LTDC IP */
    __HAL_RCC_LTDC_FORCE_RESET();
    __HAL_RCC_LTDC_RELEASE_RESET();

    /** @brief Enable the DMA2D clock */
    __HAL_RCC_DMA2D_CLK_ENABLE();

    /** @brief Toggle Sw reset of DMA2D IP */
    __HAL_RCC_DMA2D_FORCE_RESET();
    __HAL_RCC_DMA2D_RELEASE_RESET();

    /** @brief Enable DSI Host and wrapper clocks */
    __HAL_RCC_DSI_CLK_ENABLE();

    /** @brief Soft Reset the DSI Host and wrapper */
    __HAL_RCC_DSI_FORCE_RESET();
    __HAL_RCC_DSI_RELEASE_RESET();

    /** @brief NVIC configuration for LTDC interrupt that is now enabled */
    HAL_NVIC_SetPriority(LTDC_IRQn, 3, 0);
    HAL_NVIC_EnableIRQ(LTDC_IRQn);

    /** @brief NVIC configuration for DMA2D interrupt that is now enabled */
    HAL_NVIC_SetPriority(DMA2D_IRQn, 3, 0);
    HAL_NVIC_EnableIRQ(DMA2D_IRQn);

    /** @brief NVIC configuration for DSI interrupt that is now enabled */
HAL_NVIC_SetPriority(DSI_IRQn, 3, 0);
HAL_NVIC_EnableIRQ(DSI_IRQn);
}

/**
 * @brief Draws a pixel on LCD.
 * @param Xpos: X position
 * @param Ypos: Y position
 * @param RGB_Code: Pixel color in ARGB mode (8-8-8-8)
 */
void BSP_LCD_DrawPixel(uint16_t Xpos, uint16_t Ypos, uint32_t RGB_Code)
{
    /* Write data value to all SDRAM memory */
    *(volatile uint32_t*)((hltdc_discovery.LayerCfg[ActiveLayer].FBStartAdress + (4*(Ypos*BSP_LCD_GetXSize() + Xpos))) = RGB_Code;
}

/**
 * @brief Draws a character on LCD.
 * @param Xpos: Line where to display the character shape
 * @param Ypos: Start column address
 * @param c: Pointer to the character data
 */
static void DrawChar(uint16_t Xpos, uint16_t Ypos, const uint8_t *c)
{
    uint32_t i = 0, j = 0;
    uint16_t height, width;
    uint8_t offset;
    uint8_t *pchar;
    uint32_t line;
height = DrawProp[ActiveLayer].pFont->Height;
width = DrawProp[ActiveLayer].pFont->Width;
offset = 8 * ((width + 7) / 8) - width;
for (i = 0; i < height; i++)
{
    pchar = ((uint8_t *)c + (width + 7) / 8 * i);
switch(((width + 7) / 8))
{
    case 1:
        line = pchar[0];
        break;
    case 2:
        line = (pchar[0] << 8) | pchar[1];
        break;
    case 3:
    default:
        line = (pchar[0] << 16) | (pchar[1] << 8) | pchar[2];
        break;
}
for (j = 0; j < width; j++)
{
    if (line & (1 << (width - j + offset - 1)))
    {
        BSP_LCD_DrawPixel((Xpos + j), Ypos, DrawProp[ActiveLayer].TextColor);
    }
else
{
    BSP_LCD_DrawPixel((Xpos + j), Ypos, DrawProp[ActiveLayer].BackColor);
}
}
Ypos++;
}

/**
* @brief Fills a triangle (between 3 points).
* @param x1: Point 1 X position
* @param y1: Point 1 Y position
* @param x2: Point 2 X position
* @param y2: Point 2 Y position
* @param x3: Point 3 X position
* @param y3: Point 3 Y position
*/

static void FillTriangle(uint16_t x1, uint16_t x2, uint16_t x3, uint16_t y1, uint16_t y2, uint16_t y3)
{
    int16_t deltax = 0, deltay = 0, x = 0, y = 0, xinc1 = 0, xinc2 = 0,
    yinc1 = 0, yinc2 = 0, den = 0, num = 0, numadd = 0, numpixels = 0,
    curpixel = 0;

    deltax = ABS(x2 - x1);       /* The difference between the x's */
    deltay = ABS(y2 - y1);       /* The difference between the y's */
    x = x1;                      /* Start x off at the first pixel */
01797  y = y1; /* Start y off at the first pixel */
01798
01799  if (x2 >= x1) /* The x-values are increasing */
01800    {
01801      xinc1 = 1;
01802      xinc2 = 1;
01803    }
01804  else /* The x-values are decreasing */
01805    {
01806      xinc1 = -1;
01807      xinc2 = -1;
01808    }
01809
01810  if (y2 >= y1) /* The y-values are increasing */
01811    {
01812      yinc1 = 1;
01813      yinc2 = 1;
01814    }
01815  else /* The y-values are decreasing */
01816    {
01817      yinc1 = -1;
01818      yinc2 = -1;
01819    }
01820
01821  if (deltax >= deltay) /* There is at least one x-value for every y-value */
01822    {
01823      xinc1 = 0; /* Don't change the x when numerator >= denominator */
01824      yinc2 = 0; /* Don't change the y for every iteration */
01825      den = deltax;
num = deltax / 2;
numadd = deltay;
numpixels = deltax;  /* There are more x-values than y-values */
}
else  /* There is at least one y-value for every x-value */
{
    xinc2 = 0;  /* Don't change the x for every iteration */
    yinc1 = 0;  /* Don't change the y when numerator >= denominator */
    den = deltay;
    num = deltax / 2;
    numadd = deltax;
    numpixels = deltay;  /* There are more y-values than x-values */
}

for (curpixel = 0; curpixel <= numpixels; curpixel++)
{
    BSP_LCD_DrawLine(x, y, x3, y3);
    num += numadd;  /* Increase the numerator by the top of the fraction */
    if (num >= den)  /* Check if numerator >= denominator */
    {
        num -= den;  /* Calculate the new numerator value */
        x += xinc1;  /* Change the x as appropriate */
        y += yinc1;  /* Change the y as appropriate */
    }
}

x += xinc2;  /* Change the
e x as appropriate */
01852   
01853   }  
01854 }  
01855  
01856  /**
01857   * @brief Fills a buffer.
01858   * @param LayerIndex: Layer index
01859   * @param pDst: Pointer to destination buffer
01860   * @param xSize: Buffer width
01861   * @param ySize: Buffer height
01862   * @param OffLine: Offset
01863   * @param ColorIndex: Color index
01864  */
01865 static void LL_FillBuffer(uint32_t LayerIndex, void *pDst, uint32_t xSize, uint32_t ySize, uint32_t OffLine, uint32_t ColorIndex) 
01866 {  
01867   /* Register to memory mode with ARGB8888 as color Mode */
01868   hdma2d_discovery.Init.Mode = DMA2D_R2M;
01869   hdma2d_discovery.Init.ColorMode = DMA2D_OUTPUT_ARGB8888;  
01870   hdma2d_discovery.Init.OutputOffset = OffLine;
01871  
01872  
01873  
01874  /* DMA2D Initialization */
01875  if(HAL_DMA2D_Init(&hdma2d_discovery) == HAL_OK)  
01876  {
01877    if(HAL_DMA2D_ConfigLayer(&hdma2d_discovery, LayerIndex) == HAL_OK)
if (HAL_DMA2D_Start(&hdma2d_discovery, ColorIndex, (uint32_t)pDst, xSize, ySize) == HAL_OK)
{
   /* Polling For DMA transfer */
   HAL_DMA2D_PollForTransfer(&hdma2d_discovery, 10);
}

/**
 * @brief Converts a line to an ARGB8888 pixel format.
 * @param pSrc: Pointer to source buffer
 * @param pDst: Output color
 * @param xSize: Buffer width
 * @param ColorMode: Input color mode
 */
static void LL_ConvertLineToARGB8888(void *pSrc, void *pDst, uint32_t xSize, uint32_t ColorMode)
{
   /* Configure the DMA2D Mode, Color Mode and output offset */
   hdma2d_discovery.Init.Mode = DMA2D_M2M_PFC;
   hdma2d_discovery.Init.ColorMode = DMA2D_OUTPUT_ARGB8888;
   hdma2d_discovery.Init.OutputOffset = 0;
   /* Foreground Configuration */
   hdma2d_discovery.LayerCfg[1].AlphaMode = DMA2D_NO_MODIFY_ALPHA;
   hdma2d_discovery.LayerCfg[1].InputAlpha =
0xFF;
01905  hdma2d_discovery.LayerCfg[1].InputColorMode = ColorMode;
01906  hdma2d_discovery.LayerCfg[1].InputOffset = 0;
01907
01908  hdma2d_discovery.Instance = DMA2D;
01909
01910  /* DMA2D Initialization */
01911  if(HAL_DMA2D_Init(&hdma2d_discovery) == HAL_OK)
01912      {
01913       if(HAL_DMA2D_ConfigLayer(&hdma2d_discovery, 1) == HAL_OK)
01914          {
01915             if (HAL_DMA2D_Start(&hdma2d_discovery, (uint32_t)pSrc, (uint32_t)pDst, xSize, 1) == HAL_OK)
01916                {
01917                   /* Polling For DMA transfer */
01918                   HAL_DMA2D_PollForTransfer(&hdma2d_discovery, 10);
01919                }
01920          }
01921      }
01922 }
01923
01924  /**
01925  * @}}
01926  */
01927
01928  /**
01929  * @}}
01930  */
01931
01932  /**
01933  * @}
# STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td><strong>Data Structures</strong></td>
<td></td>
</tr>
</tbody>
</table>

## STM32F769I_DISCOVERY_QSPI Exported Types

STM32F769I_DISCOVERY_QSPI
Data Structures

```c
struct QSPI_Info
```

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
Go to the documentation of this file.

```c
/**
 * @file stm32f769i_discovery_qspi.h
 * @author MCD Application Team
 * @version V2.0.0
 * @date 30-December-2016
 * @brief This file contains the common defines and functions prototypes for
 * the stm32f769i_discovery_qspi.c driver.
 * @attention
 * <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
```
1. Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.
3. Neither the name of STMicroelectronics nor the names of its contributors
may be used to endorse or promote products derived from this software
without specific prior written permission.

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

/**
 * @addtogroup STM32F769I_DISCOVERY
 */

/*
 * Define to prevent recursive inclusion
 */
#ifndef __STM32F769I_DISCOVERY_QSPI_H
#define __STM32F769I_DISCOVERY_QSPI_H

#ifdef __cplusplus
extern "C"
#endif

/*
 * Includes
 */
#include "stm32f7xx_hal.h"
#include "../Components/mx25l512/mx25l512.h"

/**
 * @defgroup STM32F769I_DISCOVERY_QSPI_Exported_Constants
 */

/**
 * @defgroup STM32F769I_DISCOVERY_QSPI_Exported_Constants
 */
/* QSPI Error codes */
#define QSPI_OK ((uint8_t)0x00)
#define QSPI_ERROR ((uint8_t)0x01)
#define QSPI_BUSY ((uint8_t)0x02)
#define QSPI_NOT_SUPPORTED ((uint8_t)0x04)
#define QSPI_SUSPENDED ((uint8_t)0x08)

/* Definition for QSPI clock resources */
#define QSPI_CLK_ENABLE() __HAL_RCC_QSPI_CLK_ENABLE()
#define QSPI_CLK_DISABLE() __HAL_RCC_QSPI_CLK_DISABLE()
#define QSPI_CS_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_ENABLE()
#define QSPI_CLK_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_ENABLE()
#define QSPI_D0_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define QSPI_D1_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define QSPI_D2_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define QSPI_D3_GPIO_CLK_ENABLE() __HAL_RCC_GPIOD_CLK_ENABLE()
#define QSPI_FORCE_RESET() __HAL_RCC_QSPI_FORCE_RESET()
#define QSPI_RELEASE_RESET() __HAL_RCC_QSPI_RELEASE_RESET()

/* Definition for QSPI Pins */
#define QSPI_CS_PIN GPIO_PIN_6
```c
#define QSPI_CS_GPIO_PORT GPIOB
#define QSPI_CS_PIN_AF GPIO_AF10_QUADSPI

/* QSPI_CLK */
#define QSPI_CLK_PIN GPIO_PIN_2
#define QSPI_CLK_GPIO_PORT GPIOB
#define QSPI_CLK_PIN_AF GPIO_AF9_QUADSPI

/* QSPI_D0 */
#define QSPI_D0_PIN GPIO_PIN_9
#define QSPI_D0_GPIO_PORT GPIOC
#define QSPI_D0_PIN_AF GPIO_AF9_QUADSPI

/* QSPI_D1 */
#define QSPI_D1_PIN GPIO_PIN_10
#define QSPI_D1_GPIO_PORT GPIOC
#define QSPI_D1_PIN_AF GPIO_AF9_QUADSPI

/* QSPI_D2 */
#define QSPI_D2_PIN GPIO_PIN_2
#define QSPI_D2_GPIO_PORT GPIOE
#define QSPI_D2_PIN_AF GPIO_AF9_QUADSPI

/* QSPI_D3 */
#define QSPI_D3_PIN GPIO_PIN_13
#define QSPI_D3_GPIO_PORT GPIOD
#define QSPI_D3_PIN_AF GPIO_AF9_QUADSPI

/** *
 * @} */
```
typedef struct {
    uint32_t FlashSize; /*!< Size of the flash */
    uint32_t EraseSectorSize; /*!< Size of sectors for the erase operation */
    uint32_t EraseSectorsNumber; /*!< Number of sectors for the erase operation */
    uint32_t ProgPageSize; /*!< Size of pages for the program operation */
    uint32_t ProgPagesNumber; /*!< Number of pages for the program operation */
} QSPI_Info;

uint8_t BSP_QSPI_Init (void);
uint8_t BSP_QSPI_DeInit (void);
uint8_t BSP_QSPI_Read (uint8_t* pData, uint32_t ReadAddr, uint32_t Size);
uint8_t BSP_QSPI_Write (uint8_t* pData, uint32_t WriteAddr, uint32_t Size);
uint8_t BSP_QSPI_Erase_Block (uint32_t BlockAddress);
uint8_t BSP_QSPI_Erase_Chip (void);
uint8_t BSP_QSPI_GetStatus (void);
uint8_t BSP_QSPI_GetInfo (QSPI_Info* pInfo);
uint8_t BSP_QSPI_EnableMemoryMappedMode (void);

/* These functions can be modified in case the current settings need to be changed for specific application needs */
void BSP_QSPI_MspInit (QSPI_HandleTypeDef *hqspi, void *Params);
void BSP_QSPI_MspDeInit (QSPI_HandleTypeDef *hqspi, void *Params);

/**
 * @}
 */

/* ifndef __cplusplus */
}
#endif /* ifndef __STM32F769I_DISCOVERY_QSPI_H */
/**
 * @}
 */
#endif /* ifndef __cplusplus */
stm32f769i_discovery_qspi.c

Go to the documentation of this file.

```c
/*
 ******************************************
************************************
*/
@file stm32f769i_discovery_qspi.c
@author MCD Application Team
@version V2.0.0
@date 30-December-2016
@brief This file includes a standard driver for the MX25L512 QSPI memory mounted on STM32F769I-Discovery board.
@verbatim
==========================================
### How to use this driver ###
==========================================
```

```c
([..]
(#) This driver is used to drive the MX25L512 QSPI external memory mounted on STM32F769I-Discover
```
This driver needs a specific component driver (MX25L51245G) to be included with.

### Initialization steps:

- Initialize the QSPI external memory using the `BSP_QSPI_Init()` function. This function includes the MSP layer hardware resources initialization and the QSPI interface with the external memory.

### QSPI memory operations

- QSPI memory can be accessed with read/write operations once it is initialized.
- Read/write operation can be performed with AHB access using the functions `BSP_QSPI_Read()/BSP_QSPI_Write()`.

- The function `BSP_QSPI_GetInfo()` returns the configuration of the QSPI memory.
  (see the QSPI memory data sheet)

- Perform erase block operation using the function `BSP_QSPI_Erase_Block()` and by specifying the block address. You can perform an erase operation of the whole chip by calling the function `BSP_QSPI_Erase_Chip()`.

- The function `BSP_QSPI_GetStatus()` returns the current status of the QSPI memory.
  (see the QSPI memory data sheet)

@endverbatim

******************************************
************************************
* @attention
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

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

// Includes --------------------------------------
#include "stm32f769i_discovery_qspi.h"

/** @addtogroup BSP */
/**
 * @}
 */

/** @addtogroup STM32F769I_DISCOVERY */
/**
 * @}
 */

/** @defgroup STM32F769I_DISCOVERY_QSPI */
/**
 * @}
 */

/* Private variables --------------------------------------
-------------------------------*/

/** @defgroup STM32F769I_DISCOVERY_QSPI_Private variables */
/**
 * @}
 */
```c
/* Private functions -----------------------
----------------------------------*/

/**
 * @defgroup STM32F769I_DISCOVERY_QSPI_Private_Functions STM32F769I_DISCOVERY QSPI Private Functions
 *
 * @{*/

static uint8_t QSPI_ResetMemory(QSPI_HandleTypeDef *hqspi);
static uint8_t QSPI_EnterFourBytesAddress(QSPI_HandleTypeDef *hqspi);
static uint8_t QSPIDummyCyclesCfg(QSPI_HandleTypeDef *hqspi);
static uint8_t QSPI_EnterMemory_QPI(QSPI_HandleTypeDef *hqspi);
static uint8_t QSPI_ExitMemory_QPI(QSPI_HandleTypeDef *hqspi);
static uint8_t QSPI_OutDrvStrengthCfg(QSPI_HandleTypeDef *hqspi);
static uint8_t QSPI_WriteEnable(QSPI_HandleTypeDef *hqspi);
static uint8_t QSPI_AutoPollingMemReady(QSPI_HandleTypeDef *hqspi, uint32_t Timeout);

/* @}*/

/** @}*/
```
** @defgroup STM32F769I_DISCOVERY_QSPI_Exported_Functions STM32F769I_DISCOVERY QSPI Exported Functions

*/

/**
@brief Initializes the QSPI interface.
@retval QSPI memory status

uint8_t BSP_QSPI_Init(void)
{
    QSPIHandle.Instance = QUADSPI;

    /* Call the DeInit function to reset the driver */
    if (HAL_QSPI_DeInit(&QSPIHandle) != HAL_OK )
    {
        return QSPI_ERROR;
    }

    /* System level initialization */
    BSP_QSPI_MspInit(&QSPIHandle, NULL);

    /* QSPI initialization */
    /* QSPI freq = SYSCLK /(1 + ClockPrescaler) = 216 MHz/(1+1) = 108 Mhz */
    QSPIHandle.Init.ClockPrescaler = 1;

    /* QSPI freq = 216 MHz/(1+1) = 108 Mhz */
    QSPIHandle.Init.FifoThreshold = 16;
    QSPIHandle.Init.SampleShifting = QSPI_SAMPLE_SHIFTING_HALFCYCLE;

    QSPIHandle.Init.FlashSize = POSITIOn_VAL(MX25L512_FLASH_SIZE) - 1;
QSPIHandle.Init.ChipSelectHighTime = QSPI_CS_HIGH_TIME_4_CYCLE; /* Min 30ns for nonRead */
QSPIHandle.Init.ClockMode = QSPI_CLOCK_MODE_0;
QSPIHandle.Init.FlashID = QSPI_FLASH_ID_1;
QSPIHandle.Init.DualFlash = QSPI_DUALFLASH_DISABLE;

if (HAL_QSPI_Init(&QSPIHandle) != HAL_OK)
{
    return QSPI_ERROR;
}

/* QSPI memory reset */
if (QSPI_ResetMemory(&QSPIHandle) != QSPI_OK)
{
    return QSPI_NOT_SUPPORTED;
}

/* Put QSPI memory in QPI mode */
if( QSPI_EnterMemory_QPI( &QSPIHandle )!=QSPI_OK )
{
    return QSPI_NOT_SUPPORTED;
}

/* Set the QSPI memory in 4-bytes address mode */
if (QSPInEnterFourBytesAddress(&QSPIHandle ) != QSPI_OK)
{
    return QSPI_NOT_SUPPORTED;
}

/* Configuration of the dummy cycles on QS
/* Memory side */
00170 if (QSPI_DummyCyclesCfg(&QSPIHandle) != QS
PI_OK)
00171 {
00172     return QSPI_NOT_SUPPORTED;
00173 }
00174 /* Configuration of the Output driver strength on memory side */
00175 if (QSPI_OutDrvStrengthCfg( &QSPIHandle )
00176     != QSPI_OK )
00177 {
00178     return QSPI_NOT_SUPPORTED;
00179 }
00180 return QSPI_OK;
00181 }
00182 }/*
00183 /**
00184 * @brief De-Initializes the QSPI interface.
00185 * @retval QSPI memory status
00186 */
00187 uint8_t BSP_QSPI_DeInit(void)
00188 {
00189     QSPIHandle.Instance = QUADSPI;
00190 }
00191 /* Put QSPI memory in SPI mode */
00192 if (QSPI.ExitMemory_QPI(&QSPIHandle) != QSPI
00193     OK )
00194 {
00195     return QSPI_NOT_SUPPORTED;
00196 }
00197 /* Call the DeInit function to reset the driver */
00198 if (HAL_QSPI_DeInit(&QSPIHandle) != HAL_OK
00200  {
00201    return QSPI_ERROR;
00202  }
00203
00204  /* System level De-initialization */
00205  BSP_QSPI_MspDeInit(&QSPIHandle, NULL);
00206
00207  return QSPI_OK;
00208 }
00209
00210  /**
00211  * @brief Reads an amount of data from the QSPI memory.
00212  * @param pData: Pointer to data to be read
00213  * @param ReadAddr: Read start address
00214  * @param Size: Size of data to read
00215  * @retval QSPI memory status
00216  */
00217  uint8_t BSP_QSPI_Read(uint8_t* pData, uint32_t ReadAddr, uint32_t Size)
00218  {
00219    QSPI_CommandTypeDef s_command;
00220
00221    /* Initialize the read command */
00222    s_command.InstructionMode       = QSPI_INSTRUCTION_4_LINES;
00223    s_command.Instruction         = QPI_READ_4_BYTE_ADDR_CMD;
00224    s_command.AddressMode        = QSPI_ADDRESS_4_LINES;
00225    s_command.AddressSize       = QSPI_ADDRESS_32_BITS;
00226    s_command.Address           = ReadAddr;
00227    s_command.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE;
00228   s_command.DataMode = QSPI_DATA_4_LINES;
00229   s_command.DummyCycles = MX25L512_DUMMY_CYCLES_READ_QUAD_IO;
00230   s_command.NbData = Size;
00231   s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
00232   s_command.DdrHoldHalfCycle = QSPI_DDR_HOL_HHC_ANALOG_DELAY;
00233   s_command.SIOOMode = QSPI_SIOO_IN_ST_EVERY_CMD;
00234
00235   /* Configure the command */
00236   if (HAL_QSPI_Command(&QSPIHandle, &s_command, HAL_QSPI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00237       {
00238           return QSPI_ERROR;
00239       }
00240
00241   /* Set S# timing for Read command */
00242   MODIFY_REG(QSPIHandle.Instance->DCR, QUADSPI_DCR_CSHT, QSPI_CS_HIGH_TIME_1_CYCLE);
00243
00244   /* Reception of the data */
00245   if (HAL_QSPI_Receive(&QSPIHandle, pData, HAL_QSPI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00246       {
00247           return QSPI_ERROR;
00248       }
00249
00250   /* Restore S# timing for nonRead commands */
00251   MODIFY_REG(QSPIHandle.Instance->DCR, QUADSPI_DCR_CSHT, QSPI_CS_HIGH_TIME_4_CYCLE);
00252
00253   return QSPI_OK;
/**
 * @brief Writes an amount of data to the QSPI memory.
 * @param pData: Pointer to data to be written
 * @param WriteAddr: Write start address
 * @param Size: Size of data to write
 * @retval QSPI memory status
 */

uint8_t BSP_QSPI_Write(uint8_t* pData, uint32_t WriteAddr, uint32_t Size)
{
    QSPI_CommandTypeDef s_command;
    uint32_t end_addr, current_size, current_addr;

    /* Calculation of the size between the write address and the end of the page */
    current_size = MX25L512_PAGE_SIZE - (WriteAddr % MX25L512_PAGE_SIZE);

    /* Check if the size of the data is less than the remaining place in the page */
    if (current_size > Size)
    {
        current_size = Size;
    }

    /* Initialize the address variables */
    current_addr = WriteAddr;
    end_addr = WriteAddr + Size;

    /* Initialize the program command */
    s_command.InstructionMode = QSPI_INSTRUCTION_4_LINES;
    s_command.Instruction = QPI_PAGE_PRO
G_4_BYTE_ADDR_CMD;
00284  s_command.AddressMode = QSPI_ADDRESS _4_LINES;
00285  s_command.AddressSize = QSPI_ADDRESS _32_BITS;
00286  s_command.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE;
00287  s_command.DataMode = QSPI_DATA_4_LINES;
00288  s_command.DummyCycles = 0;
00289  s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
00290  s_command.DdrHoldHalfCycle = QSPI_DDR_HHC_ANALOG_DELAY;
00291  s_command.SIOOMode = QSPI_SIOO_INST_EVERY_CMD;
00292
00293  /* Perform the write page by page */
00294  do
00295  {
00296      s_command.Address = current_addr;
00297      s_command.NbData = current_size;
00298
00299      /* Enable write operations */
00300      if (QSPI_WriteEnable(&QSPIHandle) != QSPI_OK)
00301      {
00302          return QSPI_ERROR;
00303      }
00304
00305      /* Configure the command */
00306      if (HAL_QSPI_Command(&QSPIHandle, &s_command, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00307      {
00308          return QSPI_ERROR;
00309      }
00310  }
/* Transmission of the data */
if (HAL_QSPI_Transmit(&QSPIHandle, pData, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
  {
    return QSPI_ERROR;
  }
  
/* Configure automatic polling mode to wait for end of program */
if (QSPI_AutoPollingMemReady(&QSPIHandle, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != QSPI_OK)
  {
    return QSPI_ERROR;
  }

/* Update the address and size variables for next page programming */
current_addr += current_size;
pData += current_size;
current_size = ((current_addr + MX25L512_PAGE_SIZE) > end_addr) ? (end_addr - current_addr) : MX25L512_PAGE_SIZE;
  } while (current_addr < end_addr);
return QSPI_OK;

/**
* @brief Erases the specified block of the QSPI memory.
* @param BlockAddress: Block address to erase
* @retval QSPI memory status
*/
uint8_t BSP_QSPI_Erase_Block(uint32_t BlockAddress)
  {

QSPI_CommandTypeDef s_command;

/* Initialize the erase command */
s_command.InstructionMode = QSPI_INSTRUCTION_4_LINES;
s_command.Instruction = SUBSECTOR_ERASE_4_BYTE_ADDR_CMD;
s_command.AddressMode = QSPI_ADDRESS_4_LINES;
s_command.AddressSize = QSPI_ADDRESS_32_BITS;
s_command.Address = BlockAddress;
s_command.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE;
s_command.DataMode = QSPI_DATA_NONE;
s_command.DummyCycles = 0;
s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
s_command.DdrHoldHalfCycle = QSPI_DDR_HHC_ANALOG_DELAY;
s_command.SIOOMode = QSPI_SIOO_INST_EVERY_CMD;

/* Enable write operations */
if (QSPI_WriteEnable(&QSPIHandle) != QSPI_OK)
{
    return QSPI_ERROR;
}

/* Send the command */
if (HAL_QSPI_Command(&QSPIHandle, &s_command, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
    return QSPI_ERROR;
} /* Configure automatic polling mode to wait for end of erase */
if (QSPI_AutoPollingMemReady(&QSPIHandle, MX25L512_SUBSECTOR_ERASE_MAX_TIME) != QSPI_OK)
{
    return QSPI_ERROR;
}
return QSPI_OK;
/**
* @brief Erases the entire QSPI memory.
* @retval QSPI memory status
*/
uint8_t BSP_QSPI_Erase_Chip(void)
{
    QSPI_CommandTypeDef s_command;
    /* Initialize the erase command */
    s_command.InstructionMode = QSPI_INSTRUCTION_4_LINES;
    s_command.Instruction = BULK_ERASE_CMD;
    s_command.AddressMode = QSPI_ADDRESS_NONE;
    s_command.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE;
    s_command.DataMode = QSPI_DATA_NONE;
    s_command.DummyCycles = 0;
    s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
    s_command.DdrHoldHalfCycle = QSPI_DDR_HHC_ANALOG_DELAY;
s_command.SIOOMode = QSPI_SIOO_IN
ST_EVERY_CMD;

/* Enable write operations */
if (QSPI_WriteEnable(&QSPIHandle) != QSPI_OK)
{
    return QSPI_ERROR;
}

/* Send the command */
if (HAL_QSPI_Command(&QSPIHandle, &s_command, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
    return QSPI_ERROR;
}

/* Configure automatic polling mode to wait for end of erase */
if (QSPI_AutoPollingMemReady(&QSPIHandle, MX25L512_BULK_ERASE_MAX_TIME) != QSPI_OK)
{
    return QSPI_ERROR;
}

return QSPI_OK;

/**
 * @brief Reads current status of the QSPI memory.
 * @retval QSPI memory status */
uint8_t BSP_QSPI_GetStatus(void)
{
    QSPI_CommandTypeDef s_command;
    uint8_t reg;
/* Initialize the read flag status register command */
s_command.InstructionMode = QSPI_INSTRUCTION_4_LINES;
s_command.Instruction = READ_STATUS_REG_CMD;
s_command.AddressMode = QSPI_ADDRESS_NONE;
s_command.AlternateByteMode = QSPI_ALTERNATEBYTES_NONE;
s_command.DataMode = QSPI_DATA_4_LINES;
s_command.DummyCycles = 0;
s_command.NbData = 1;
s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
s_command.DdrHoldHalfCycle = QSPI_DDR_HHC_ANALOG_DELAY;
s_command.SIOOMode = QSPI_SIOO_INST_EVERY_CMD;

/* Configure the command */
if (HAL_QSPI_Command(&QSPIHandle, &s_command, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
    return QSPI_ERROR;
}

/* Reception of the data */
if (HAL_QSPI_Receive(&QSPIHandle, &reg, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
    return QSPI_ERROR;
}

/* Check the value of the register*/
if ((reg & MX25L512_SR_WIP) == 0) {
    return QSPI_OK;
} else {
    return QSPI_BUSY;
}

/**
  * @brief  Return the configuration of the QSPI memory.
  * @param  pInfo: pointer on the configuration structure
  * @retval QSPI memory status
  */
uint8_t BSP_QSPI_GetInfo(QSPI_Info* pInfo)
{
    /* Configure the structure with the memory configuration */
    pInfo->FlashSize = MX25L512_FLASH_SIZE;
    pInfo->EraseSectorSize = MX25L512_SUBSECTOR_SIZE;
    pInfo->EraseSectorsNumber = (MX25L512_FLASH_SIZE/MX25L512_SUBSECTOR_SIZE);
    pInfo->ProgPageSize = MX25L512_PAGE_SIZE;
    pInfo->ProgPagesNumber = (MX25L512_FLASH_SIZE/MX25L512_PAGE_SIZE);
    return QSPI_OK;
}

/**
  * @brief  Configure the QSPI in memory-map
  */
ped mode
00478  * @retval QSPI memory status
00479  */
00480 uint8_t BSP_QSPI_EnableMemoryMappedMode(void)
00481 {
00482  QSPI_CommandTypeDef s_command;
00483  QSPI_MemoryMappedTypeDef s_mem_mapped_cfg;
00484  /* Configure the command for the read instruction */
00485  s_command.InstructionMode = QSPI_INSTRUCTION_4_LINES;
00486  s_command.Instruction = QSPI_READ_4_BYTE_ADDR_CMD;
00487  s_command.AddressMode = QSPI_ADDRESS_4_LINES;
00488  s_command.AddressSize = QSPI_ADDRESS_32_BITS;
00489  s_command.AlternateByteMode = QSPI_ALTERNATEBYTES_NONE;
00490  s_command.DataMode = QSPI_DATA_4_LINES;
00491  s_command.DummyCycles = MX25L512_DUMMY_CYCLES_READ_QUAD_IO;
00492  s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
00493  s_command.DdrHoldHalfCycle = QSPI_DDR_HHC_ANALOG_DELAY;
00494  s_command.SIOOMode = QSPI_SIO_ON_IDLE_DISABLE;
00495  /* Configure the memory mapped mode */
00496  s_mem_mapped_cfg.TimeOutActivation = QSPI_TIMEOUT_COUNTER_DISABLE;
00497  s_mem_mapped_cfg.TimeOutPeriod = 0;
if (HAL_QSPI_MemoryMapped(&QSPIHandle, &s_command, &s_mem_mapped_cfg) != HAL_OK)
{
    return QSPI_ERROR;
}

return QSPI_OK;

/**
 * @}
 */

/**
 * @addtogroup STM32F769I_DISCOVERY_QSPI_Private_Functions
 * @{
 */

/**
 * @brief QSPI MSP Initialization
 * This function configures the hardware resources used in this example:
 * - Peripheral's clock enable
 * - Peripheral's GPIO Configuration
 * - NVIC configuration for QSPI interrupt
 * @retval None
 */

__weak void BSP_QSPI_MspInit(QSPI_HandleTypeDef *hqspi, void *Params)
{
    GPIO_InitTypeDef gpio_init_structure;

    /*##-1- Enable peripherals and GPIO Clocks

    ###################################################################*/

    /* Enable the QuadSPI memory interface clo
ck */
00531     QSPI_CLK_ENABLE();
00532     /* Reset the QuadSPI memory interface */
00533     QSPI_FORCE_RESET();
00534     QSPI_RELEASE_RESET();
00535     /* Enable GPIO clocks */
00536     QSPI_CS_GPIO_CLK_ENABLE();
00537     QSPI_CLK_GPIO_CLK_ENABLE();
00538     QSPI_D0_GPIO_CLK_ENABLE();
00539     QSPI_D1_GPIO_CLK_ENABLE();
00540     QSPI_D2_GPIO_CLK_ENABLE();
00541     QSPI_D3_GPIO_CLK_ENABLE();
00542
00543    /*###-2- Configure peripheral GPIO #######

00544    */
00545    /* QSPI CS GPIO pin configuration */
00546    gpio_init_structure.Pin = QSPI_CS_PIN;
00547    gpio_init_structure.Alternate = QSPI_CS_PIN_AF;
00548    gpio_init_structure.Mode = GPIO_MODE_AF_PP;
00549    gpio_init_structure.Pull = GPIO_PULLUP;
00550    gpio_init_structure.Speed = GPIO_SPEED_FREQ_HIGH;
00551    HAL_GPIO_Init(QSPI_CS_GPIO_PORT, &gpio_init_structure);
00552    /* QSPI CLK GPIO pin configuration */
00553    gpio_init_structure.Pin = QSPI_CLK_PIN;
00554    gpio_init_structure.Alternate = QSPI_CLK_PIN_AF;
00555    gpio_init_structure.Pull = GPIO_NOPULL;
00556    HAL_GPIO_Init(QSPI_CLK_GPIO_PORT, &gpio_init_structure);
/* QSPI D0 GPIO pin configuration */
gpio_init_structure.Pin = QSPI_D0_PIN;
gpio_init_structure.Alternate = QSPI_D0_PIN_AF;
HAL_GPIO_Init(QSPI_D0_GPIO_PORT, &gpio_init_structure);

/* QSPI D1 GPIO pin configuration */
gpio_init_structure.Pin = QSPI_D1_PIN;
gpio_init_structure.Alternate = QSPI_D1_PIN_AF;
HAL_GPIO_Init(QSPI_D1_GPIO_PORT, &gpio_init_structure);

/* QSPI D2 GPIO pin configuration */
gpio_init_structure.Pin = QSPI_D2_PIN;
gpio_init_structure.Alternate = QSPI_D2_PIN_AF;
HAL_GPIO_Init(QSPI_D2_GPIO_PORT, &gpio_init_structure);

/* QSPI D3 GPIO pin configuration */
gpio_init_structure.Pin = QSPI_D3_PIN;
gpio_init_structure.Alternate = QSPI_D3_PIN_AF;
HAL_GPIO_Init(QSPI_D3_GPIO_PORT, &gpio_init_structure);

/*##-3- Configure the NVIC for QSPI ######
###################################*/
HAL_NVIC_SetPriority(QUADSPI_IRQn, 0x0F, 0);
HAL_NVIC_EnableIRQ(QUADSPI_IRQn);

/* NVIC configuration for QSPI interrupt */
HAL_NVIC_SetPriority(QUADSPI_IRQn, 0x0F, 0);
HAL_NVIC_EnableIRQ(QUADSPI_IRQn);
/**
 * @brief QSPI MSP De-Initialization
 * This function frees the hardware resources used in this example:
 * - Disable the Peripheral's clock
 * - Revert GPIO and NVIC configuration to their default state
 * @retval None
 */

__weak void BSP_QSPI_MspDeInit(QSPI_HandleTypeDef *hqspi, void *Params)
{
    /*##-1- Disable the NVIC for QSPI #/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/*/
    HAL_NVIC_DisableIRQ(QUADSPI_IRQn);

    /*##-2- Disable peripherals and GPIO Clocks #/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/#/
/* Disable the QuadSPI memory interface clock */
QSPI_CLK_DISABLE();
}

/**
 * @brief This function resets the QSPI memory.
 * @param hqspi: QSPI handle
 * @retval None
 */
static uint8_t QSPI_ResetMemory(QSPI_HandleTypeDef *hqspi)
{
    QSPI_CommandTypeDef s_command;
    QSPI_AutoPollingTypeDef s_config;
    uint8_t reg;

    /* Send command RESET command in QPI mode (QUAD I/Os) */
    /* Initialize the reset enable command */
    s_command.InstructionMode = QSPI_INSTRUCTION_4_LINES;
    s_command.Instruction = RESET_ENABLE_CMD;
    s_command.AddressMode = QSPI_ADDRESS_NONE;
    s_command.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE;
    s_command.DataMode = QSPI_DATA_NONE;
    s_command.DummyCycles = 0;
    s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
if (HAL_QSPI_Command(hqspi, &s_command, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
    return QSPI_ERROR;
}

/* Send the reset memory command */
s_command.Instruction = RESET_MEMORY_CMD;
if (HAL_QSPI_Command(hqspi, &s_command, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
    return QSPI_ERROR;
}

/* Send command RESET command in SPI mode */
/* Initialize the reset enable command */
s_command.InstructionMode = QSPI_INSTRUCTION_1_LINE;
s_command.Instruction = RESET_ENABLE_CMD;
if (HAL_QSPI_Command(hqspi, &s_command, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
    return QSPI_ERROR;
}

/* Send the reset memory command */
s_command.Instruction = RESET_MEMORY_CMD;
if (HAL_QSPI_Command(hqspi, &s_command, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
    return QSPI_ERROR;
}
/* After reset CMD, 1000ms requested if QSPI memory SWReset occurred during full chip erase operation */
HAL_Delay( 1000 );
/* Configure automatic polling mode to wait the WIP bit=0 */
s_config.Match = 0;
s_config.Mask = MX25L512_SR_WIP;
s_config.MatchMode = QSPI_MATCH_MODE_AND;
s_config>StatusBytesSize = 1;
s_config.Interval = 0x10;
s_config.AutomaticStop = QSPI_AUTOMATIC_STOP_ENABLE;
s_command.InstructionMode = QSPI_INSTRUCTION_1_LINE;
s_command/Instruction = READ_STATUS_REG_CMD;
s_command.DataMode = QSPI_DATA_1_LINE;
if (HAL_QSPI_AutoPolling(hqspi, &s_command, &s_config, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
    return QSPI_ERROR;
}
/* Initialize the reading of status register */
s_command.InstructionMode = QSPI_INSTRUCTION_1_LINE;
s_command.Instruction = READ_STATUS_REG_CMD;
s_command.AddressMode = QSPI_ADDRESS_NONE;
s_command.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE;
s_command.DataMode = QSPI_DATA_1_LINE;
s_command.DummyCycles = 0;
s_command.NbData = 1;
s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
s_command.DdrHoldHalfCycle = QSPI_DDR_HHC_ANALOG_DELAY;
s_command.SIOOMode = QSPI_SIOO_INST_EVERY_CMD;

/* Configure the command */
if (HAL_QSPI_Command(hqspi, &s_command, HAL_QSPI_TIMEOUT_DEFAULT_VALUE) != HAL_OK) {
    return QSPI_ERROR;
}

/* Reception of the data */
if (HAL_QSPI_Receive(hqspi, &reg, HAL_QSPI_TIMEOUT_DEFAULT_VALUE) != HAL_OK) {
    return QSPI_ERROR;
}

/* Enable write operations, command in 1 bit */
/* Enable write operations */
s_command.InstructionMode = QSPI_INSTRUCTION_1_LINE;
s_command.Instruction = WRITE_ENABLE
CMD;
00707 s_command.AddressMode = QSPI_ADDRESS_NONE;
00708 s_command.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE;
00709 s_command.DataMode = QSPI_DATA_NONE;
00710 s_command.DummyCycles = 0;
00711 s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
00712 s_command.DdrHoldHalfCycle = QSPI_DDR_HHANALOG_DELAY;
00713 s_command.SIOOMode = QSPI_SIOO_INST_EVERY_CMD;
00714 00715 if (HAL_QSPI_Command(hqspi, &s_command, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00716     { 00717         return QSPI_ERROR;
00718     }
00719 00720 /* Configure automatic polling mode to wait for write enabling */
00721 s_config.Match = MX25L512_SR_WREN;
00722 s_config.Mask = MX25L512_SR_WREN;
00723 s_config.MatchMode = QSPI_MATCH_MODE_AND;
00724 s_config.StatusBytesSize = 1;
00725 s_config.Interval = 0x10;
00726 s_config.AutomaticStop = QSPI_AUTOMATIC_STOP_ENABLE;
00727 00728 s_command.Instruction = READ_STATUS_REG_CMD;
00729 s_command.DataMode = QSPI_DATA_1_LIN
if (HAL_QSPI_AutoPolling(hqspi, &s_command, &s_config, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
    return QSPI_ERROR;
}

/* Update the configuration register with new dummy cycles */
s_command.InstructionMode = QSPI_INSTRUCTION_1_LINE;
s_command.InstructionCFG_REG_CMD = WRITE_STATUS_CFG_REG_CMD;
s_command.AddressMode = QSPI_ADDRESS_NONE;
s_command.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE;
s_command.DataMode = QSPI_DATA_1_LINE;
s_command.DummyCycles = 0;
s_command.NbData = 1;
s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
s_command.DdrHoldHalfCycle = QSPI_DDR_HHC_ANALOG_DELAY;
s_command.SIOOMode = QSPI_SIOO_INST_EVERY_CMD;

/* Enable the Quad IO on the QSPI memory (Non-volatile bit) */
reg |= MX25L512_SR_QUADEN;

/* Configure the command */
if (HAL_QSPI_Command(hqspi, &s_command, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{  
  return QSPI_ERROR;
}

/* Transmission of the data */
if (HAL_QSPI_Transmit(hqspi, &reg, HAL_QSPI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
  return QSPI_ERROR;
}

/* 40ms Write Status/Configuration Register Cycle Time */
HAL_Delay( 40 );
return QSPI_OK;

/**
 * @brief This function set the QSPI memory in 4-byte address mode
 * @param hqspi: QSPI handle
 * @retval None
 */
static uint8_t QSPI_EnterFourBytesAddress(QSPI_HandleTypeDef *hqspi)
{
  QSPI_CommandTypeDef s_command;

  /* Initialize the command */
  s_command.InstructionMode = QSPI_INSTRUCTION_4_LINES;
  s_command.Instruction = ENTER_4_BYTE_ADDR_MODE_CMD;
  s_command.AddressMode = QSPI_ADDRESS_NONE;
  s_command.AlternateByteMode = QSPI_ALTERNATEBYTE_NONE;
  return QSPI_OK;
}
TE_BYTES_NONE;
00783    s_command.DataMode       = QSPI_DATA_NONE;
00784    s_command.DummyCycles   = 0;
00785    s_command.DdrMode       = QSPI_DDR_MODE_DISABLE;
00786    s_command.DdrHoldHalfCycle = QSPI_DDR_HHC_ANALOG_DELAY;
00787    s_command.SIOOMode     = QSPI_SIOO_INSTIT_EVERY_CMD;
00788
00789    /* Enable write operations */
00790    if (QSPI_WriteEnable(hqspi) != QSPI_OK)
00791    {
00792       return QSPI_ERROR;
00793    }
00794
00795    /* Send the command */
00796    if (HAL_QSPI_Command(hqspi, &s_command, HAL_QSPI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00797    {
00798       return QSPI_ERROR;
00799    }
00800
00801    /* Configure automatic polling mode to wait the memory is ready */
00802    if (QSPI_AutoPollingMemReady(hqspi, HAL_QSPI_TIMEOUT_DEFAULT_VALUE) != QSPI_OK)
00803    {
00804       return QSPI_ERROR;
00805    }
00806
00807    return QSPI_OK;
00808 }
00809
00810 /**
00811    * @brief This function configure the dumm
static uint8_t QSPI_DummyCyclesCfg(QSPI_HandleTypeDef *hqspi) {
    QSPI_CommandTypeDef s_command;
    uint8_t reg[2];

    /* Initialize the reading of status register */
    s_command.InstructionMode = QSPI_INSTRUCTION_4_LINES;
    s_command.Instruction = READ_STATUS_REG_CMD;
    s_command.AddressMode = QSPI_ADDRESS_NONE;
    s_command.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE;
    s_command.DataMode = QSPI_DATA_4_LINES;
    s_command.DummyCycles = 0;
    s_command.NbData = 1;
    s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
    s_command.DdrHoldHalfCycle = QSPI_DDR_HHC_Analog_Delay;
    s_command.SIOOMode = QSPI_SIOO_INST_EVERY_CMD;

    /* Configure the command */
    if (HAL_QSPI_Command(hqspi, &s_command, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK) {
        return QSPI_ERROR;
    }
}

/* Reception of the data */
if (HAL_QSPI_Receive(hqspi, &(reg[0]), HAL_QSPI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
    return QSPI_ERROR;
}

/* Initialize the reading of configuration register */
s_command.InstructionMode = QSPI_INSTRUCTION_4_LINES;
s_command.Instruction = READ_CFG_REG_CMD;
s_command.AddressMode = QSPI_ADDRESS_NONE;
s_command.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE;
s_command.DataMode = QSPI_DATA_4_LINES;
s_command.DummyCycles = 0;
s_command.NbData = 1;
s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
s_command.DdrHoldHalfCycle = QSPI_DDR_HHC_ANALOG_DELAY;
s_command.SIOOMode = QSPI_SIOO_INST_EVERY_CMD;

/* Configure the command */
if (HAL_QSPI_Command(hqspi, &s_command, HAL_QSPI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
    return QSPI_ERROR;
}

/* Reception of the data */
if ((HAL_QSPI_Receive(hqspi, &reg[1]), HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK) {
    return QSPI_ERROR;
}

/* Enable write operations */
if (QSPI_WriteEnable(hqspi) != QSPI_OK) {
    return QSPI_ERROR;
}

/* Update the configuration register with new dummy cycles */
s_command.InstructionMode = QSPI_INSTRUCTION_4_LINES;
s_command.Instruction = WRITE_STATUS_CFG_REG_CMD;
s_command.AddressMode = QSPI_ADDRESS_NONE;
s_command.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE;
s_command.DataMode = QSPI_DATA_4_LINES;
s_command.DummyCycles = 0;
s_command.NbData = 2;
s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
s_command.DdrHoldHalfCycle = QSPI_DDR_HHC_ANALOG_DELAY;
s_command.SIOOMode = QSPI_SIOO_INST_EVERY_CMD;

/* MX25L512_DUMMY_CYCLES_READ QUAD = 3 for 10 cycles in QPI mode */
MODIFY_REG( reg[1], MX25L512_CR_NB_DUMMY, (MX25L512_DUMMY_CYCLES_READ_QUAD << POSITION_VAL(M)})
/* Configure the write volatile configuration register command */
if (HAL_QSPI_Command(hqspi, &s_command, HAL_QSPI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
    return QSPI_ERROR;
}

/* Transmission of the data */
if (HAL_QSPI_Transmit(hqspi, &(reg[0]), HAL_QSPI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
    return QSPI_ERROR;
}

/* 40ms Write Status/Configuration Register Cycle Time */
HAL_Delay(40);
return QSPI_OK;

/**
 * @brief This function put QSPI memory in QPI mode (quad I/O).
 * @param hqspi: QSPI handle
 * @retval None
 */
static uint8_t QSPI_EnterMemory_QPI(QSPI_HandleTypeDef *hqspi)
{
    QSPI_CommandTypeDef s_command;
    QSPI_AutoPollingTypeDef s_config;
    /* Initialize the QPI enable command */
/* QSPI memory is supported to be in SPI mode, so CMD on 1 LINE */
s_command.InstructionMode = QSPI_INSTRUCTION_1_LINE;
s_command.Instruction = ENTER_QUAD_CMD;
s_command.AddressMode = QSPI_ADDRESS_NONE;
s_command.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE;
s_command.DataMode = QSPI_DATA_NONE;
s_command.DummyCycles = 0;
s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
s_command.DdrHoldHalfCycle = QSPI_DDR_HHC_ANALOG_DELAY;
s_command.SIOOMode = QSPI_SIOO_INST_EVERY_CMD;

/* Send the command */
if (HAL_QSPI_Command(hqspi, &s_command, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
    return QSPI_ERROR;
}

/* Configure automatic polling mode to wait the QUADEN bit=1 and WIP bit=0 */
s_config.Match = MX25L512_SR_QUADEN;
s_config.Mask = MX25L512_SR_QUADEN|MX25L512_SR_WIP;
s_config.MatchMode = QSPI_MATCH_MODE_AND;
s_config.StatusBytesSize = 1;
s_config.Interval = 0x10;
s_config.AutomaticStop = QSPI_AUTOMATIC_STOP_ENABLE;
s_command.InstructionMode = QSPI_INSTRUCTION_4_LINES;
s_command.Instruction = READ_STATUS_REG_CMD;
s_command.DataMode = QSPI_DATA_4_LINES;

if (HAL_QSPI_AutoPolling(hqspi, &s_command, &s_config, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
    return QSPI_ERROR;
}

return QSPI_OK;

/**
 * @brief This function put QSPI memory in SPI mode.
 * @param hqspi: QSPI handle
 * @retval None
 */
static uint8_t QSPI_ExitMemory_QPI(QSPI_HandleTypeDef *hqspi)
{
    QSPI_CommandTypeDef s_command;

    /* Initialize the QPI enable command */
    /* QSPI memory is supported to be in QPI mode, so CMD on 4 LINES */
    s_command.InstructionMode = QSPI_INSTRUCTION_4_LINES;
    s_command.Instruction = EXIT_QUAD_CM
D;
00968  s_command.AddressMode     = QSPI_ADDRESS_NONE;
00969  s_command.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE;
00970  s_command.DataMode       = QSPI_DATA_NONE;
00971  s_command.DummyCycles    = 0;
00972  s_command.DdrMode        = QSPI_DDR_MODE_DISABLE;
00973  s_command.DdrHoldHalfCycle = QSPI_DDR_HOLD_HALF_CYCLE_ANALOG_DELAY;
00974  s_command.SIOOMode       = QSPI_SIOO_INSTIT_EVERY_CMD;
00975
00976  /* Send the command */
00977  if (HAL_QSPI_Command(hqspi, &s_command, HAL_QSPI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00978      {
00979      return QSPI_ERROR;
00980  }
00981
00982  return QSPI_OK;
00983 }
00984
00985  /**
00986  * @brief This function configure the Output driver strength on memory side.
00987  * @param hqspi: QSPI handle
00988  * @retval None
00989  */
00990 static uint8_t QSPI_OutDrvStrengthCfg( QSPI_HandleTypeDef *hqspi )
00991  {
00992  QSPI_CommandTypeDef s_command;
00993  uint8_t reg[2];
00994
/* Initialize the reading of status register */
s_command.InstructionMode = QSPI_INSTRUCTION_4_LINES;
s_command.Instruction = READ_STATUS_REG_CMD;
s_command.AddressMode = QSPI_ADDRESS_NONE;
s_command.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE;
s_command.DataMode = QSPI_DATA_4_LINES;
s_command.DummyCycles = 0;
s_command.NbData = 1;
s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
s_command.DdrHoldHalfCycle = QSPI_DDR_HHC_ANALOG_DELAY;
s_command.SIOOMode = QSPI_SIOO_INST_EVERY_CMD;

/* Configure the command */
if (HAL_QSPI_Command(hqspi, &s_command, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
  return QSPI_ERROR;
}

/* Reception of the data */
if (HAL_QSPI_Receive(hqspi, &(reg[0]), HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
  return QSPI_ERROR;
}

/* Initialize the reading of configuration register */
01020  s_command.InstructionMode = QSPI_INSTRUCTION_4_LINES;
01021  s_command.Instruction  = READ_CFG_REG_CMD;
01022  s_command.AddressMode  = QSPI_ADDRESS_NONE;
01023  s_command.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE;
01024  s_command.DataMode = QSPI_DATA_4_LINES;
01025  s_command.DummyCycles = 0;
01026  s_command.NbData = 1;
01027  s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
01028  s_command.DdrHoldHalfCycle = QSPI_DDR_HHC_ANALOG_DELAY;
01029  s_command.SIOOMode = QSPI_SIOO_INST_EVERY_CMD;
01030
01031  /* Configure the command */
01032  if (HAL_QSPI_Command(hqspi, &s_command, HAL_QSPI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
01033      {
01034          return QSPI_ERROR;
01035      }
01036
01037  /* Reception of the data */
01038  if (HAL_QSPI_Receive(hqspi, &(reg[1]), HAL_QSPI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
01039      {
01040          return QSPI_ERROR;
01041      }
01042
01043  /* Enable write operations */
01044  if (QSPI_WriteEnable(&QSPIHandle) != QSPI_OK)
01045      {

01046    return QSPI_ERROR;
01047    }
01048
01049    /* Update the configuration register with new output driver strength */
01050    s_command.InstructionMode = QSPI_INSTRUCTION_4_LINES;
01051    s_command.Instruction = WRITE_STATUS_CFG_REG_CMD;
01052    s_command.AddressMode = QSPI_ADDRESS_NONE;
01053    s_command.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE;
01054    s_command.DataMode = QSPI_DATA_4_LINES;
01055    s_command.DummyCycles = 0;
01056    s_command.NbData = 2;
01057    s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
01058    s_command.DdrHoldHalfCycle = QSPI_DDR_HHC_ANALOG_DELAY;
01059    s_command.SIOOMode = QSPI_SIOO_INST_EVERY_CMD;
01060
01061    /* Set Output Strength of the QSPI memory 15 ohms */
01063
01064    /* Configure the write volatile configuration register command */
01065    if (HAL_QSPI_Command(hqspi, &s_command, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
01066    {
01067        return QSPI_ERROR;
01068    }
/* Transmission of the data */
if (HAL_QSPI_Transmit(hqspi, &(reg[0]), HAL_QSPI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
{
    return QSPI_ERROR;
}

return QSPI_OK;

/**
 * @brief This function send a Write Enable and wait it is effective.
 * @param hqspi: QSPI handle
 * @retval None
 */
static uint8_t QSPI_WriteEnable(QSPI_HandleTypeDef *hqspi)
{
    QSPI_CommandTypeDef s_command;
    QSPI_AutoPollingTypeDef s_config;

    /* Enable write operations */
    s_command.InstructionMode = QSPI_INSTRUCTION_4_LINES;
    s_command.Instruction = WRITE_ENABLE_CMD;
    s_command.AddressMode = QSPI_ADDRESS_NONE;
    s_command.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE;
    s_command.DataMode = QSPI_DATA_NONE;
    s_command.DummyCycles = 0;
    s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
    s_command.DdrHoldHalfCycle = QSPI_DDR_HHC...
if (HAL_QSPI_Command(hqspi, &s_command, HAL_QSPI_TIMEOUT_DEFAULT_VALUE) != HAL_OK) {
    return QSPI_ERROR;
}

/* Configure automatic polling mode to wait for write enabling */
s_config.Match = MX25L512_SR_WREN;
s_config.Mask = MX25L512_SR_WREN;
s_config.MatchMode = QSPI_MATCH_MODE_AND;
s_config.StatusBytesSize = 1;
s_config.Interval = 0x10;
s_config.AutomaticStop = QSPI_AUTOMATIC_STOP_ENABLE;

if (HAL_QSPI_AutoPolling(hqspi, &s_command, &s_config, HAL_QSPI_TIMEOUT_DEFAULT_VALUE) != HAL_OK) {
    return QSPI_ERROR;
}

return QSPI_OK;
/**
 * @brief This function read the SR of the memory and wait the EOP.
 * @param hqspi: QSPI handle
 * @param Timeout
 * @retval None
 */

static uint8_t QSPI_AutoPollingMemReady(QSPI_HandleTypeDef *hqspi, uint32_t Timeout)
{
    QSPI_CommandTypeDef s_command;
    QSPI_AutoPollingTypeDef s_config;

    /* Configure automatic polling mode to wait for memory ready */
    s_command.InstructionMode = QSPI_INSTRUCTION_4_LINES;
    s_command.Instruction = READ_STATUS_REG_CMD;
    s_command.AddressMode = QSPI_ADDRESS_NONE;
    s_command.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE;
    s_command.DataMode = QSPI_DATA_4_LINES;
    s_command.DummyCycles = 0;
    s_command.DdrMode = QSPI_DDR_MODE_DISABLE;
    s_command.DdrHoldHalfCycle = QSPI_DDR_HHC_ANALOG_DELAY;
    s_command.SIOOMode = QSPI_SIOO_INST_EVERY_CMD;

    s_config.Match = 0;
    s_config.Mask = MX25L512_SR_WIP;
}
s_config.MatchMode = QSPI_MATCH_MODE
_AND;
s_config.StatusBytesSize = 1;
s_config.Interval = 0x10;
s_config.AutomaticStop = QSPI_AUTOMATIC_STOP_ENABLE;

if (HAL_QSPI_AutoPolling(hqspi, &s_command, &s_config, Timeout) != HAL_OK)
{
    return QSPI_ERROR;
}

return QSPI_OK;

 /**<
 * @}*/
 /**<
 * @}*/
 /**<
 * @}*/
 /**<
 * @}*/

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE*****/
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td>Modules</td>
<td>Defines</td>
</tr>
</tbody>
</table>

**STM32F769I_DISCOVERY SD**

---

STM32F769I_DISCOVERY
### Modules

<table>
<thead>
<tr>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F769I Discovery Sd Private TypesDef</td>
</tr>
<tr>
<td>STM32F769I Discovery Sd Private Defines</td>
</tr>
<tr>
<td>STM32F769I Discovery Sd Private Macro</td>
</tr>
<tr>
<td>STM32F769I Discovery Sd Private Variables</td>
</tr>
<tr>
<td>STM32F769I Discovery Sd Private Prototypes</td>
</tr>
<tr>
<td>STM32F769I Discovery Sd Private Functions</td>
</tr>
<tr>
<td>SD Exported Types</td>
</tr>
<tr>
<td>SD Exported Constants</td>
</tr>
<tr>
<td>STM32F769I Discovery SD Exported Macro</td>
</tr>
<tr>
<td>STM32F769I Discovery SD Exported Functions</td>
</tr>
</tbody>
</table>
## Defines

<table>
<thead>
<tr>
<th>define</th>
<th>value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSD_OK</td>
<td>((uint8_t)0x00)</td>
<td>SD status structure definition.</td>
</tr>
<tr>
<td>MSD_ERROR</td>
<td>((uint8_t)0x01)</td>
<td></td>
</tr>
<tr>
<td>MSD_ERROR_SD_NOT_PRESENT</td>
<td>((uint8_t)0x02)</td>
<td></td>
</tr>
<tr>
<td>SD_TRANSFER_OK</td>
<td>((uint8_t)0x00)</td>
<td>SD transfer state definition.</td>
</tr>
<tr>
<td>SD_TRANSFER_BUSY</td>
<td>((uint8_t)0x01)</td>
<td></td>
</tr>
</tbody>
</table>
Define Documentation

```c
#define MSD_ERROR   ((uint8_t)0x01)
```

Definition at line 78 of file `stm32f769i_discovery_sd.h`.

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

```c
#define MSD_ERROR_SD_NOT_PRESENT   ((uint8_t)0x02)
```

Definition at line 79 of file `stm32f769i_discovery_sd.h`.

Referenced by `BSP_SD_Init()`.

```c
#define MSD_OK     ((uint8_t)0x00)
```

SD status structure definition.

Definition at line 77 of file `stm32f769i_discovery_sd.h`.

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

```c
#define SD_TRANSFER_BUSY   ((uint8_t)0x01)
```

Definition at line 85 of file `stm32f769i_discovery_sd.h`.

Referenced by `BSP_SD_GetCardState()`.
#define SD_TRANSFER_OK ((uint8_t)0x00)

SD transfer state definition.

Definition at line 84 of file stm32f769i_discovery_sd.h.

Referenced by BSP_SD_GetCardState().
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>STM32F769I_DISCOVERY SDRAM</strong></td>
<td>Modules</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>STM32F769I_DISCOVERY</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Modules

<table>
<thead>
<tr>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDRAM Private Types Definitions</td>
</tr>
<tr>
<td>SDRAM Private Defines</td>
</tr>
<tr>
<td>SDRAM Private Macros</td>
</tr>
<tr>
<td>SDRAM Private Variables</td>
</tr>
<tr>
<td>SDRAM Private Function Prototypes</td>
</tr>
<tr>
<td>SDRAM Private Functions</td>
</tr>
<tr>
<td>SDRAM Exported Types</td>
</tr>
<tr>
<td>SDRAM Exported Macro</td>
</tr>
<tr>
<td>SDRAM Exported Functions</td>
</tr>
</tbody>
</table>

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>STM32F769I_DISCOVERY TS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>STM32F769I_DISCOVERY</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Modules

<table>
<thead>
<tr>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS Private Types Definitions</td>
</tr>
<tr>
<td>TS Private Types Defines</td>
</tr>
<tr>
<td>TS Private Macros</td>
</tr>
<tr>
<td>TS Imported Variables</td>
</tr>
<tr>
<td>TS Private Variables</td>
</tr>
<tr>
<td>TS Private Function Prototypes</td>
</tr>
<tr>
<td>TS Public Functions</td>
</tr>
<tr>
<td>TS Exported Constants</td>
</tr>
<tr>
<td>TS Exported Types</td>
</tr>
<tr>
<td>TS Exported Functions</td>
</tr>
</tbody>
</table>

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TS Exported Types**

STM32F769I_DISCOVERY TS
Data Structures

<table>
<thead>
<tr>
<th>struct</th>
<th>TS_StateTypeDef</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS_StateTypeDef</td>
<td>Define TS State structure. More...</td>
</tr>
</tbody>
</table>
### Enumerations

<table>
<thead>
<tr>
<th>Enum Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TS_StatusTypeDef</strong></td>
<td>Define BSP_TS_xxx() functions possible return value, when status is returned by those functions. More...</td>
</tr>
<tr>
<td></td>
<td><strong>TS_GestureIdTypeDef</strong></td>
</tr>
<tr>
<td></td>
<td><strong>TS_TouchEventTypeDef</strong></td>
</tr>
</tbody>
</table>

**TS_StatusTypeDef**

```c
enum TS_StatusTypeDef { TS_OK = 0x00, TS_ERROR = 0x01, TS_TIMEOUT = 0x02, TSDEVICE_NOT_FOUND = 0x03 }
```

**TS_GestureIdTypeDef**

```c
enum TS_GestureIdTypeDef {
    GEST_ID_NO_GESTURE = 0x00,
    GEST_ID_MOVE_UP = 0x01,
    GEST_ID_MOVE_RIGHT = 0x02,
    GEST_ID_MOVE_DOWN = 0x03,
    GEST_ID_MOVE_LEFT = 0x04,
    GEST_ID_ZOOM_IN = 0x05,
    GEST_ID_ZOOM_OUT = 0x06,
    GEST_ID_NB_MAX = 0x07
}
```

**TS_TouchEventTypeDef**

```c
enum TS_TouchEventTypeDef {
    TOUCH_EVENT_NO_EVT = 0x00,
    TOUCH_EVENT_PRESS_DOWN = 0x01,
    TOUCH_EVENT_LIFT_UP = 0x02,
    TOUCH_EVENT_CONTACT = 0x03,
    TOUCH_EVENT_NB_MAX = 0x04
}
```
Enumeration Type Documentation

enum TS_GestureIdTypeDef

TS_GestureIdTypeDef Define Possible managed gesture identification values returned by touch screen driver.

**Enumerator:**

- `GEST_ID_NO_GESTURE` Gesture not defined / recognized
- `GEST_ID_MOVE_UP` Gesture Move Up
- `GEST_ID_MOVE_RIGHT` Gesture Move Right
- `GEST_ID_MOVE_DOWN` Gesture Move Down
- `GEST_ID_MOVE_LEFT` Gesture Move Left
- `GEST_ID_ZOOM_IN` Gesture Zoom In
- `GEST_ID_ZOOM_OUT` Gesture Zoom Out
- `GEST_ID_NB_MAX` max number of gesture id

Definition at line 125 of file `stm32f769i_discovery_ts.h`.

enum TS_StatusTypeDef
TS_StatusTypeDef Define BSP_TS_xxx() functions possible return value, when status is returned by those functions.

**Enumerator:**

- **TS_OK**  Touch Ok
- **TS_ERROR**  Touch Error
- **TS_TIMEOUT**  Touch Timeout
- **TS_DEVICE_NOT_FOUND**  Touchscreen device not found

Definition at line 112 of file stm32f769i_discovery_ts.h.

```c
enum TS_TouchEventTypeDef
```

TS_TouchEventTypeDef Define Possible touch events kind as returned values by touch screen IC Driver.

**Enumerator:**

- **TOUCH_EVENT_NO_EVT**  Touch Event : undetermined
- **TOUCH_EVENT_PRESS_DOWN**  Touch Event Press Down
- **TOUCH_EVENT_LIFT_UP**  Touch Event Lift Up
- **TOUCH_EVENT_CONTACT**  Touch Event Contact
**TOUCH_EVENT_NB_MAX**  
max number of touch events kind

Definition at line 142 of file `stm32f769i_discovery_ts.h`.

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by [doxygen](https://www.doxygen.org/) 1.7.6.1
stm32f769i_discovery_ts.h

Go to the documentation of this file.

```c
/**
 ******************************************
 ************************************
 * @file stm32f769i_discovery_ts.h
 * @author MCD Application Team
 * @version V2.0.0
 * @date 30-December-2016
 * @brief This file contains the common defines and functions prototypes for
 *        the stm32f769i_discovery_ts.c driver.
 ******************************************
 ************************************
 *
 * @attention

 * <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>

 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
```
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

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

#ifndef __STM32F769I_DISCOVERY_TS_H
#define __STM32F769I_DISCOVERY_TS_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes -----------------------------------*/
#include "stm32f769i_discovery.h"
#include "stm32f769i_discovery_lcd.h"

/* Include TouchScreen component driver */
#include "../Components/ft6x06/ft6x06.h"

/** @addtogroup BSP *
 * @{
 */

/** @addtogroup STM32F769I_DISCOVERY *
 * @{
 */

/** @defgroup STM32F769I_DISCOVERY_TS STM32F769I_DISCOVERY TS *
 * @{
 */

/** @defgroup STM32F769I_DISCOVERY_TS_Exported_Constants TS Exported Constants */

/** @addtogroup BSP *
 * @{
 */

/** @addtogroup STM32F769I_DISCOVERY *
 * @{
 */

/** @defgroup STM32F769I_DISCOVERY_TS STM32F769I_DISCOVERY TS *
 * @{
 */

/** @defgroup STM32F769I_DISCOVERY_TS_Exported_Constants TS Exported Constants */

/** @} */

/* @} */

#ifdef __cplusplus
}
#endif
#endif
*/
/**
 * @brief With FT6206 : maximum 2 touches detected simultaneously
 */
#define TS_MAX_NB_TOUCH (uint32_t) FT6206_MAX_DETECTABLE_TOUCH

#define TS_NO_IRQ_PENDING (uint8_t) 0
#define TS_IRQ_PENDING (uint8_t) 1

#define TS_SWAP_NONE (uint8_t) 0x01
#define TS_SWAP_X (uint8_t) 0x02
#define TS_SWAP_Y (uint8_t) 0x04
#define TS_SWAP_XY (uint8_t) 0x08

/**
 * @defgroup STM32F769I_DISCOVERY_TS_Exported_Types TS Exported Types
 */

/**
 * @defgroup TS_StateTypeDef
 * Define TS State structure
 */
typedef struct {
    uint8_t touchDetected; /*!
} TS_StateTypeDef; /*!
< Total number of active touches detected at last scan */
00095   uint16_t touchX[TS_MAX_NB_TOUCH];       /*!
00096   uint16_t touchY[TS_MAX_NB_TOUCH];       /*!
< Touch X[0], X[1] coordinates on 12 bits */
00097   #if (TS_MULTI_TOUCH_SUPPORTED == 1)
00098       uint8_t touchWeight[TS_MAX_NB_TOUCH]; /*!
< Touch_Weight[0], Touch_Weight[1] : weight property of touches */
00100       uint8_t touchEventId[TS_MAX_NB_TOUCH];
< Touch_EventId[0], Touch_EventId[1] : take value of type @ref TS_TouchEventTypeDef */
00101       uint8_t touchArea[TS_MAX_NB_TOUCH];  /*!
< Touch_Area[0], Touch_Area[1] : touch area of each touch */
00102       uint32_t gestureId;                  /*!< type of gesture detected : take value of type @ref TS_GestureIdType Def */
00103   #endif  /* TS_MULTI_TOUCH_SUPPORTED == 1 */
00104
00105 } TS_StateTypeDef;
00106
00107 /**
00108   * @brief TS_StatusTypeDef
00109   * Define BSP_TS_xxx() functions possible return value,
00110   * when status is returned by those functions.
00111   */
00112 typedef enum
00113   {
00114       TS_OK = 0x00,                  /*!< Touch Ok */
00115       TS_ERROR = 0x01,               /*!< Touch Error */
TS_TIMEOUT = 0x02,/*!< Touch Timeout */
TS_DEVICE_NOT_FOUND = 0x03/*!< Touchscreen device not found */
}

/**
 * @brief TS_GestureIdTypeDef
 * Define Possible managed gesture identification values returned by touch screen
driver.
 */

typedef enum
{
  GEST_ID_NO_GESTURE = 0x00,/*!< Gesture not defined / recognized */
  GEST_ID_MOVE_UP = 0x01,/*!< Gesture Move Up */
  GEST_ID_MOVE_RIGHT = 0x02,/*!< Gesture Move Right */
  GEST_ID_MOVE_DOWN = 0x03,/*!< Gesture Move Down */
  GEST_ID_MOVE_LEFT = 0x04,/*!< Gesture Move Left */
  GEST_ID_ZOOM_IN = 0x05,/*!< Gesture Zoom In */
  GEST_ID_ZOOM_OUT = 0x06,/*!< Gesture Zoom Out */
  GEST_ID_NB_MAX = 0x07/*!< max number of gesture id */
} TS_GestureIdTypeDef;

/**
 * @brief TS_TouchEventTypeDef
 * Define Possible touch events kind as returned values
 * by touch screen IC Driver.
 */
typedef enum {
    TOUCH_EVENT_NO_EVT = 0x00, /*!< Touch Event : undetermined */
    TOUCH_EVENT_PRESS_DOWN = 0x01, /*!< Touch Event Press Down */
    TOUCH_EVENT_LIFT_UP = 0x02, /*!< Touch Event Lift Up */
    TOUCH_EVENT_CONTACT = 0x03, /*!< Touch Event Contact */
    TOUCH_EVENT_NB_MAX = 0x04 /*!< max number of touch events kind */
} TS_TouchEventTypeDef;

/**
 * @addtogroup STM32F769I_DISCOVERY_TS_Imported_Variables
 * @{
 *
 * @brief Table for touchscreen event information display on LCD:
 * table indexed on enum @ref TS_TouchEventTypeDef information
 *
 * extern char * ts_event_string_tab[TOUCH_EVENT_NB_MAX];
 *
 * @brief Table for touchscreen gesture Id information display on LCD: table indexed
 * on enum @ref TS_GestureIdTypeDef information
 */
extern char * ts_gesture_id_string_tab[GEST_ID_NB_MAX];

/*
 * @}
 */

/**
 * @defgroup STM32F769I_DISCOVERY_TS_Exported_Functions TS Exported Functions
 * @{
 */

uint8_t BSP_TS_Init(uint16_t ts_SizeX, uint16_t ts_SizeY);
uint8_t BSP_TS_GetState(TS_StateTypeDef *TS_State);

#if (TS_MULTI_TOUCH_SUPPORTED == 1)
uint8_t BSP_TS_Get_GestureId(TS_StateTypeDef *TS_State);
uint8_t BSP_TS_ResetTouchData(TS_StateTypeDef *TS_State);
#endif /* TS_MULTI_TOUCH_SUPPORTED == 1 */

uint8_t BSP_TS_ITConfig(void);

/* These __weak function can be surcharged by application code in case the current settings need to be changed for specific (example GPIO allocation) */
void BSP_TS_INT_MspInit(void);

/*
 * @}
 */

*/

/*
 * @}
 */
/**
   * @}
*/

#ifdef __cplusplus
}
#endif

#endif /*__STM32F769I_DISCOVERY_TS_H*/

/**************************************************************************	(C)	COPYRIGHT	STMicroelectronics	*****END	OF	FILE****/
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TS Exported Constants**

STM32F769I_DISCOVERY TS
Defines

<table>
<thead>
<tr>
<th>#define</th>
<th>TS_MAX_NB_TOUCH</th>
<th>((uint32_t) FT6206_MAX_DETECTABLE TOUCH)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>With FT6206: maximum 2 touches detected simultaneously.</td>
<td></td>
</tr>
<tr>
<td>#define</td>
<td>TS_NO_IRQ_PENDING</td>
<td>((uint8_t) 0)</td>
</tr>
<tr>
<td>#define</td>
<td>TS_IRQ_PENDING</td>
<td>((uint8_t) 1)</td>
</tr>
<tr>
<td>#define</td>
<td>TS_SWAP_NONE</td>
<td>((uint8_t) 0x1)</td>
</tr>
<tr>
<td>#define</td>
<td>TS_SWAP_X</td>
<td>((uint8_t) 0x2)</td>
</tr>
<tr>
<td>#define</td>
<td>TS_SWAP_Y</td>
<td>((uint8_t) 0x4)</td>
</tr>
<tr>
<td>#define</td>
<td>TS_SWAP_XY</td>
<td>((uint8_t) 0x8)</td>
</tr>
</tbody>
</table>
Define Documentation

#define TS_IRQ_PENDING  ((uint8_t) 1)

Definition at line 74 of file stm32f769i_discovery_ts.h.

#define TS_MAX_NB_TOUCH  ((uint32_t) FT6206_MAX_DETECTABLE_TOUCH)

With FT6206: maximum 2 touches detected simultaneously.

Definition at line 71 of file stm32f769i_discovery_ts.h.

Referenced by BSP_TS_GetState().

#define TS_NO_IRQ_PENDING  ((uint8_t) 0)

Definition at line 73 of file stm32f769i_discovery_ts.h.

#define TS_SWAP_NONE  ((uint8_t) 0x01)

Definition at line 76 of file stm32f769i_discovery_ts.h.

Referenced by BSP_TS_Init().

#define TS_SWAP_X  ((uint8_t) 0x02)

Definition at line 77 of file stm32f769i_discovery_ts.h.

Referenced by BSP_TS_GetState().

#define TS_SWAP_XY  ((uint8_t) 0x08)
Definition at line 79 of file `stm32f769i_discovery_ts.h`.

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

```c
#define TS_SWAP_Y ((uint8_t) 0x04)
```

Definition at line 78 of file `stm32f769i_discovery_ts.h`.

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

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery
BSP User Manual by `doxygen` 1.7.6.1
stm32f769i_discovery_ts.c

Go to the documentation of this file.

```c
/**
 * @file    stm32f769i_discovery_ts.c
 * @author  MCD Application Team
 * @version V2.0.0
 * @date    30-December-2016
 * @brief   This file provides a set of functions needed to manage the Touch Screen on STM32F769I-DISCOVERY discovery board.
 */

/* Copyright(c) 2016 STM Microelectronics */

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

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
 CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
1. How To use this driver:
- This driver is used to drive the touch screen module of the STM32F769I-DISCOVERY discovery unit board on the K.O.D Optica Technology 480x800 TFT-LCD mounted on MB1166 daughter board. The touch screen driver IC inside the K.O.D module KM-040TMP-02 is a FT6206 by Focal Tech.

2. Driver description:
+ Initialization steps:
- Initialize the TS module using the BSP_TS_Init() function. This function includes the MSP layer hardware resources initialization and the communication layer configuration to start the TS use. The LCD size properties (x and y) are passed as parameters.
- If TS interrupt mode is desired, you must configure the TS interrupt mode by calling the function BSP_TS_ITConfig(). The TS interrupt mode is generated as an external interrupt whenever a touch is detected.
- The interrupt mode internally uses the IO functionalities driver driven by
the IO expander, to configure the IT line.

+ Touch screen use
  The touch screen state is captured whenever the function BSP_TS_GetState() is used. This function returns information about the last LCD touch occurred in the TS_StateTypeDef structure.
  The IT is handled using the corresponding external interrupt IRQ handler, the user IT callback treatment is implemented on the same external interrupt callback.

--------------------------------------------
/* Includes ----------------------------------*/
#include "stm32f769i_discovery.h"
#include "stm32f769i_discovery_ts.h"

/** @addtogroup BSP */
* @{
** @addtogroup STM32F769I_DISCOVERY */
* @{
** @defgroup STM32F769I_DISCOVERY_TS */
* @}
/**	@defgroup STM32F769I_DISCOVERY_TS_Private_Defines TS Private Types Defines
*/

/**	@defgroup STM32F769I_DISCOVERY_TS_Private_Macros TS Private Macros
*/

/**	@defgroup STM32F769I_DISCOVERY_TS_Imported_Variables TS Imported Variables
*/

/**	@defgroup STM32F769I_DISCOVERY_TS_Private_Variables TS Private Variables
*/

static TS_DrvTypeDef *ts_driver;
static uint8_t ts_orientation;
uint8_t I2C_Address = 0;

/* Table for touchscreen event information display on LCD : table indexed on enum @ref TS_Touch
EventTypeDef information */
char * ts_event_string_tab[TOUCH_EVENT_NB_MAX] = { "None",
    "Press down",
    "Lift up",
    "Contact"
};

/* Table for touchscreen gesture Id information display on LCD : table indexed on enum @ref TS
_GestureIdTypeDef information */
char * ts_gesture_id_string_tab[GEST_ID_NB_MAX] = { "None",
    "Move Up",
    "Move Right",
    "Move Down",
    "Move Left",
    "Zoom In",
    "Zoom Out"
};
/**	@defgroup STM32F769I_DISCOVERY_TS_Private_Function_Prototypes TS Private Function Prototypes
*/

/**	@defgroup STM32F769I_DISCOVERY_TS_Public_Functions TS Public Functions
*/

/**	@brief Initializes and configures the touch screen functionalities and configures all necessary hardware resources (GPIOs, I2C, clocks..).
* @param ts_SizeX : Maximum X size of the TS area on LCD
* @param ts_SizeY : Maximum Y size of the TS area on LCD
* @retval TS_OK if all initializations are OK. Other value if error.
*/
uint8_t BSP_TS_Init(uint16_t ts_SizeX, uint16_t ts_SizeY)
{
    uint8_t ts_status = TS_OK;
    uint8_t ts_id1, ts_id2 = 0;
    /* Note : I2C_Address is un-initialized here, but is not used at all in init function */
/* but the prototype of Init() is like that in template and should be respected */

/* Initialize the communication channel to sensor (I2C) if necessary */
/* that is initialization is done only once after a power up */

_ft6x06_ts_drv.Init(I2C_Address);

ts_id1 = _ft6x06_ts_drv.ReadID(TS_I2C_ADDRESS);

if(ts_id1 != FT6206_ID_VALUE)
{
    ts_id2 = _ft6x06_ts_drv.ReadID(TS_I2C_ADDRESS_A02);
    I2C_Address = TS_I2C_ADDRESS_A02;
}
else
{
    I2C_Address = TS_I2C_ADDRESS;
}

/* Scan FT6xx6 TouchScreen IC controller ID register by I2C Read */
/* Verify this is a FT6206 or FT6336G, otherwise this is an error case */
if((ts_id1 == FT6206_ID_VALUE) || (ts_id2 == FT6206_ID_VALUE))
{
    /* Found FT6206 : Initialize the TS driver structure */
    ts_driver = &_ft6x06_ts_drv;

    /* Get LCD chosen orientation */
    if(ts_SizeX < ts_SizeY)
    {
        ts_orientation = TS_SWAP_NONE;
}  
else  
{
    ts_orientation = TS_SWAP_XY | TS_SWAP_Y;
}  

if (ts_status == TS_OK)
{
    /* Software reset the TouchScreen */
    ts_driver->Reset(I2C_Address);

    /* Calibrate, Configure and Start the TouchScreen driver */
    ts_driver->Start(I2C_Address);
}

} /* of if(ts_status == TS_OK) */

}  
else  
{
    ts_status = TS_DEVICE_NOT_FOUND;
}  

return (ts_status);

/**
 * @brief Configures and enables the touch screen interrupts.
 * @retval TS_OK if all initializations are OK. Other value if error.
 */

uint8_t BSP_TS_ITConfig(void)  
{
    uint8_t ts_status = TS_OK;
    GPIO_InitTypeDef gpio_init_structure;

/* Msp Init of GPIO used for TS_INT pin coming from TouchScreen driver IC FT6x06 */
/* When touchscreen is operated in interrupt mode */

BSP_TS_INT_MspInit();

/* Configure Interrupt mode for TS_INT pin falling edge : when a new touch is available */
/* TS_INT pin is active on low level on new touch available */
gpio_init_structure.Pin = TS_INT_PIN;
gpio_init_structure.Pull = GPIO_PULLUP;
gpio_init_structure.Speed = GPIO_SPEED_FAST;
gpio_init_structure.Mode = GPIO_MODE_IT_FALLING;
HAL_GPIO_Init(TS_INT_GPIO_PORT, &gpio_init_structure);

/* Enable and set the TS_INT EXTI Interrupt to an intermediate priority */
HAL_NVIC_SetPriority((IRQn_Type)(TS_INT_EXTI_IRQn), 0x0F, 0x00);
HAL_NVIC_EnableIRQ((IRQn_Type)(TS_INT_EXTI_IRQn));

/* Enable the TS in interrupt mode */
/* In that case the INT output of FT6206 when new touch is available */
/* is active on low level and directed on EXTI */
ts_driver->EnableIT(I2C_Address);

return (ts_status);

/**
 * @brief Returns status and positions of the touch screen.
 * @param TS_State: Pointer to touch screen current state structure
 * @retval TS_OK if all initializations are OK. Other value if error.
 */

uint8_t BSP_TS_GetState(TS_StateTypeDef *TS_State) {
    static uint32_t _x[TS_MAX_NB_TOUCH] = {0, 0};
    static uint32_t _y[TS_MAX_NB_TOUCH] = {0, 0};
    uint8_t ts_status = TS_OK;
    uint16_t tmp;
    uint16_t Raw_x[TS_MAX_NB_TOUCH];
    uint16_t Raw_y[TS_MAX_NB_TOUCH];
    uint16_t xDiff;
    uint16_t yDiff;
    uint32_t index;
    #if (TS_MULTI_TOUCH_SUPPORTED == 1)
        uint32_t weight = 0;
        uint32_t area = 0;
        uint32_t event = 0;
    #endif
    /* Check and update the number of touches active detected */
    TS_State->touchDetected = ts_driver->DetectTouch(I2C_Address);
    if(TS_State->touchDetected) {
        for(index=0; index < TS_State->touchDetected; index++) {
            // Further code...
        }
    }
}
/* Get each touch coordinates */

if (ts_orientation & TS_SWAP_XY)
{
    tmp = Raw_x[index];
    Raw_x[index] = Raw_y[index];
    Raw_y[index] = tmp;
}

if (ts_orientation & TS_SWAP_X)
{
    Raw_x[index] = FT_6206_MAX_WIDTH - 1 - Raw_x[index];
}

if (ts_orientation & TS_SWAP_Y)
{
    Raw_y[index] = FT_6206_MAX_HEIGHT - 1 - Raw_y[index];
}

xDiff = Raw_x[index] > _x[index]? (Raw_x[index] - _x[index]): (_x[index] - Raw_x[index])
;
yDiff = Raw_y[index] > _y[index]? (Raw_y[index] - _y[index]): (_y[index] - Raw_y[index])
;
if (((xDiff + yDiff) > 5))
{
    _x[index] = Raw_x[index];
    _y[index] = Raw_y[index];
}
00310     TS_State->touchX[index] = _x[index];
00311     TS_State->touchY[index] = _y[index];
00312
00313     #if (TS_MULTI_TOUCH_SUPPORTED == 1)
00314
00315        /* Get touch info related to the current touch */
00316        ft6x06_TS_GetTouchInfo(I2C_Address, index, &weight, &area, &event);
00317
00318        /* Update TS_State structure */
00319        TS_State->touchWeight[index] = weight;
00320        TS_State->touchArea[index] = area;
00321
00322        /* Remap touch event */
00323        switch(event)
00324        {
00325            case FT6206 TOUCH_EVT_FLAG_PRESS_DOWN :
00326                TS_State->touchEventId[index] = TOUCHEVENT_PRESS_DOWN;
00327                break;
00328            case FT6206 TOUCH_EVT_FLAG_LIFT_UP :
00329                TS_State->touchEventId[index] = TOUCHEVENT_LIFT_UP;
00330                break;
00331            case FT6206 TOUCH_EVT_FLAG_CONTACT :
00332                TS_State->touchEventId[index] = TOUCHEVENT_CONTACT;
00333                break;
00334            case FT6206 TOUCH_EVT_FLAG_NO_EVENT :
00335                TS_State->touchEventId[index] = TOUCHEVENT_NO_EVT;
00336                break;
00337        default :
00338                ts_status = TS_ERROR;
break;
} /* of switch(event) */
#endif /* TS_MULTI_TOUCH_SUPPORTED == 1 */

/*	of	switch(event)	*/

#if (TS_MULTI_TOUCH_SUPPORTED == 1)
/*	Get gesture Id */
ts_status = BSP_TS_Get_GestureId(TS_State);
#endif /* TS_MULTI_TOUCH_SUPPORTED == 1 */

} /* end of if(TS_State->touchDetected != 0) */

return (ts_status);
}

#if (TS_MULTI_TOUCH_SUPPORTED == 1)
/**
 * @brief Update gesture Id following a touch detected.
 * @param TS_State: Pointer to touch screen current state structure
 * @retval TS_OK if all initializations are OK. Other value if error.
 */
uint8_t BSP_TS_Get_GestureId(TS_StateTypeDef *TS_State)
{
uint32_t gestureId = 0;
uint8_t ts_status = TS_OK;

/* Get gesture Id */
ft6x06_TS_GetGestureID(I2C_Address, &gestu
reId);
/* Remap gesture Id to a TS_GestureIdTypeD ef value */
switch(gestureId) {
    case FT6206_GEST_ID_NO_GESTURE :
        TS_State->gestureId = GEST_ID_NO_GESTURE;
        break;
    case FT6206_GEST_ID_MOVE_UP :
        TS_State->gestureId = GEST_ID_MOVE_UP;
        break;
    case FT6206_GEST_ID_MOVE_RIGHT :
        TS_State->gestureId = GEST_ID_MOVE_RIGHT;
        break;
    case FT6206_GEST_ID_MOVE_DOWN :
        TS_State->gestureId = GEST_ID_MOVE_DOWN;
        break;
    case FT6206_GEST_ID_MOVE_LEFT :
        TS_State->gestureId = GEST_ID_MOVE_LEFT;
        break;
    case FT6206_GEST_ID_ZOOM_IN :
        TS_State->gestureId = GEST_ID_ZOOM_IN;
        break;
    case FT6206_GEST_ID_ZOOM_OUT :
        TS_State->gestureId = GEST_ID_ZOOM_OUT;
        break;
    default :
        ts_status = TS_ERROR;
        break;
} /* of switch(gestureId) */
return(ts_status);
#endif /* TS_MULTI_TOUCH_SUPPORTED == 1 */

/**	@defgroup STM32F769I_DISCOVERY_TS_Private_Functions TS Private Functions
* @{
*/
#if (TS_MULTI_TOUCH_SUPPORTED == 1)
/**
* @brief Function used to reset all touch data before a new acquisition of touch information.
* @param TS_State: Pointer to touch screen current state structure
* @retval TS_OK if OK, TE_ERROR if problem found.
*/
uint8_t BSP_TS_ResetTouchData(TS_StateTypeDef *TS_State)
{
    uint8_t ts_status = TS_ERROR;
    uint32_t index;

    if (TS_State != (TS_StateTypeDef *)NULL)
    {
        TS_State->gestureId = GEST_ID_NO_GESTURE;
        TS_STATE->touchDetected = 0;
        for(index = 0; index < TS_MAX_NB_TOUCH; index++)
        {
            TS_STATE->touchX[index] = 0;
            TS_STATE->touchY[index] = 0;
        }
    }

```
TS_State->touchArea[index] = 0;
TS_State->touchEventId[index] = TOUCH_EVENT_NO_EVT;
TS_State->touchWeight[index] = 0;
}

ts_status = TS_OK;
}
/*	of	if	(TS_State	!=	(TS_StateTypeDef	*
)
#endif /* TS_MULTI_TOUCH_SUPPORTED == 1 */
/**
@brief	Initializes the TS_INT pin MSP.
@retval	None
*/
__weak void BSP_TS_INT_MspInit(void)
{
    GPIO_InitTypeDef gpio_init_structure;
    TS_INT_GPIO_CLK_ENABLE();
    /* GPIO configuration in input for TouchScreen interrupt signal on TS_INT pin */
gpio_init_structure.Pin = TS_INT_PIN;
gpio_init_structure.Mode = GPIO_MODE_INPUT;
gpio_init_structure.Pull = GPIO_PULLUP;
gpio_init_structure.Speed = GPIO_SPEED_HIGH;
HAL_GPIO_Init(TS_INT_GPIO_PORT, &gpio_init
_structure);
00459 }
00460 
00461 /**
00462   * @}
00463 */
00464 
00465 /**
00466   * @}
00467 */
00468 
00469 /**
00470   * @}
00471 */
00472 
00473 /**
00474   * @}
00475 */
00476 
00477 /**
00478   * @}
00479 */
00480 
00481 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
STM32F769I-Discovery BSP User Manual

STM32F769I_DISCOVERY EEPROM

This file includes the I2C EEPROM driver of STM32F769I-DISCOVERY board. More...
### Modules

<table>
<thead>
<tr>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEPROM Private Types</td>
</tr>
<tr>
<td>EEPROM Private Defines</td>
</tr>
<tr>
<td>EEPROM Private Macros</td>
</tr>
<tr>
<td>EEPROM Private Variables</td>
</tr>
<tr>
<td>EEPROM Private Function Prototypes</td>
</tr>
<tr>
<td>EEPROM Private Functions</td>
</tr>
<tr>
<td>EEPROM Exported Types</td>
</tr>
<tr>
<td>EEPROM Exported Constants</td>
</tr>
<tr>
<td>EEPROM Exported Macros</td>
</tr>
<tr>
<td>EEPROM Exported Functions</td>
</tr>
</tbody>
</table>
Detailed Description

This file includes the I2C EEPROM driver of STM32F769I-DISCOVERY board.
## SDRAM Exported Constants

**SDRAM Exported Types**
Defines

```c
#define SDRAM_DEVICE_ADDR ((uint32_t)0xC0000000) /* SDRAM device address */
#define SDRAM_DEVICE_SIZE ((uint32_t)0x1000000) /* SDRAM device size in MBytes */
#define SDRAM_MEMORY_WIDTH FMC_SDRAM_MEM_BUS_WIDTH_32
#define SDCLOCK_PERIOD FMC_SDRAM_CLOCK_PERIOD_2
#define REFRESH_COUNT ((uint32_t)0x0603) /* SDRAM refresh count (100Mhz SD clock) */
#define SDRAM_TIMEOUT ((uint32_t)0xFFFF)
#define __DMAx_CLK_ENABLE __HAL_RCC_DMA2_CLK_ENABLE
#define __DMAx_CLK_DISABLE __HAL_RCC_DMA2_CLK_DISABLE
#define SDRAM_DMAx_CHANNEL DMA_CHANNEL_0
#define SDRAM_DMAx_STREAM DMA2_Stream0
#define SDRAM_DMAx_IRQn DMA2_Stream0_IRQn
#define BSP_SDRAM_DMA_IRQHandler DMA2_Stream0_IRQHandler
```
Define Documentation

```c
#define __DMAx_CLK_DISABLE __HAL_RCC_DMA2_CLK_DISABLE

Definition at line 92 of file stm32f769i_discovery_sdram.h.

#define __DMAx_CLK_ENABLE __HAL_RCC_DMA2_CLK_ENABLE

Definition at line 91 of file stm32f769i_discovery_sdram.h.

Referenced by BSP_SDRAM_MspInit().

#define BSP_SDRAM_DMA_IRQHandler DMA2_Stream0_IRQHandler

Definition at line 96 of file stm32f769i_discovery_sdram.h.

#define REFRESH_COUNT ((uint32_t)0x0603) /* SDRAM refresh counter (100Mhz SD clock)*/

Definition at line 86 of file stm32f769i_discovery_sdram.h.

Referenced by BSP_SDRAM_Init().

#define SDCLOCK_PERIOD FMC_SDRAM_CLOCK_PERIOD_2

Definition at line 83 of file stm32f769i_discovery_sdram.h.

Referenced by BSP_SDRAM_Init().

#define SDRAM_DEVICE_ADDR ((uint32_t)0xC0000000)

Definition at line 76 of file stm32f769i_discovery_sdram.h.
```
#define SDRAM_DEVICE_SIZE  ((uint32_t)0x1000000) /* SDRAM device size in MBytes */

Definition at line 77 of file stm32f769i_discovery_sdram.h.

#define SDRAM_DMAx_CHANNEL DMA_CHANNEL_0

Definition at line 93 of file stm32f769i_discovery_sdram.h.

Referenced by BSP_SDRAM_MspInit().

#define SDRAM_DMAx_IRQn DMA2_Stream0_IRQHandler

Definition at line 95 of file stm32f769i_discovery_sdram.h.

Referenced by BSP_SDRAM_MspDeInit(), and
BSP_SDRAM_MspInit().

#define SDRAM_DMAx_STREAM DMA2_Stream0

Definition at line 94 of file stm32f769i_discovery_sdram.h.

Referenced by BSP_SDRAM_MspDeInit(), and
BSP_SDRAM_MspInit().

#define SDRAM_MEMORY_WIDTH FMC_SDRAM_MEM_BUS_WIDTH_32

Definition at line 81 of file stm32f769i_discovery_sdram.h.

Referenced by BSP_SDRAM_Init().

#define SDRAM_TIMEOUT  ((uint32_t)0xFFFF)
Definition at line 88 of file stm32f769i_discovery_sdram.h.

Referenced by BSP_SDRAM_Initialization_sequence(), and BSP_SDRAM_Sendcmd().
# STM32F769I-Discovery BSP User Manual

## Directories

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## SD Exported Constants

`STM32F769I_DISCOVERY_SD`
### Defines

```c
#define SD_PRESENT ((uint8_t)0x01)
#define SD_NOT_PRESENT ((uint8_t)0x00)
#define SD_DATETIMEOUT ((uint32_t)100000000)
#define __DMAx_TxRx_CLK_ENABLE __HAL_RCC_DMA2_CLK_ENABLE
#define SD_DMAx_Tx_CHANNEL DMA_CHANNEL_11
#define SD_DMAx_Rx_CHANNEL DMA_CHANNEL_11
#define SD_DMAx_Tx_STREAM DMA2_Stream5
#define SD_DMAx_Rx_STREAM DMA2_Stream0
#define SD_DMAx_Tx_IRQn DMA2_Stream5_IRQn
#define SD_DMAx_Rx_IRQn DMA2_Stream0_IRQn
#define BSP_SDMMC_IRQHandler SDMMC2_IRQHandler
#define BSP_SDMMC_DMA_Tx_IRQHandler DMA2_Stream5_IRQHandler
#define BSP_SDMMC_DMA_Rx_IRQHandler DMA2_Stream0_IRQHandler
#define SD_DetectIRQHandler() HAL_GPIO_EXTI_IRQHandler(SD_DETECT_PIN)
```
Define Documentation

```cpp
#define __DMAx_TxRx_CLK_ENABLE __HAL_RCC_DMA2_CLK_ENABLE

Definition at line 96 of file stm32f769i_discovery_sd.h.
Referenced by BSP_SD_MspInit().

#define BSP_SDMMC_DMA_Rx_IRQHandler DMA2_Stream0_IRQHandler

Definition at line 105 of file stm32f769i_discovery_sd.h.

#define BSP_SDMMC_DMA_Tx_IRQHandler DMA2_Stream5_IRQHandler

Definition at line 104 of file stm32f769i_discovery_sd.h.

#define BSP_SDMMC_IRQHandler SDMMC2_IRQHandler

Definition at line 103 of file stm32f769i_discovery_sd.h.

#define SD_DATATIMEOUT ((uint32_t)100000000)

Definition at line 93 of file stm32f769i_discovery_sd.h.

#define SD_DetectIRQHandler ( ) HAL_GPIO_EXTI_IRQHandler(SD_DETECT_PIN)

Definition at line 106 of file stm32f769i_discovery_sd.h.

#define SD_DMAx_Rx_CHANNEL DMA_CHANNEL_11
```
Definition at line 98 of file stm32f769i_discovery_sd.h.
Referenced by BSP_SD_MspInit().

```c
#define SD_DMAx_Rx_IRQHandler DMA2_Stream0_IRQHandler
```
Definition at line 102 of file stm32f769i_discovery_sd.h.
Referenced by BSP_SD_MspDeInit(), and BSP_SD_MspInit().

```c
#define SD_DMAx_Rx_STREAM DMA2_Stream0
```
Definition at line 100 of file stm32f769i_discovery_sd.h.
Referenced by BSP_SD_MspDeInit(), and BSP_SD_MspInit().

```c
#define SD_DMAx.Tx_CHANNEL DMA_CHANNEL_11
```
Definition at line 97 of file stm32f769i_discovery_sd.h.
Referenced by BSP_SD_MspInit().

```c
#define SD_DMAx.Tx_IRQHandler DMA2_Stream5_IRQHandler
```
Definition at line 101 of file stm32f769i_discovery_sd.h.
Referenced by BSP_SD_MspDeInit(), and BSP_SD_MspInit().

```c
#define SD_DMAx.Tx_STREAM DMA2_Stream5
```
Definition at line 99 of file stm32f769i_discovery_sd.h.
Referenced by BSP_SD_MspDeInit(), and BSP_SD_MspInit().
```c
#define SD_NOT_PRESENT ((uint8_t)0x00)
```
Definition at line 91 of file `stm32f769i_discovery_sd.h`.
Referenced by `BSP_SD_IsDetected()`.

```c
#define SD_PRESENT ((uint8_t)0x01)
```
Definition at line 90 of file `stm32f769i_discovery_sd.h`.
Referenced by `BSP_SD_Init()`, and `BSP_SD_IsDetected()`.
## LOW_LEVEL Private Defines

| STM32F769I_DISCOVERY LOW LEVEL | Defines |
## Defines

<table>
<thead>
<tr>
<th>Define</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__STM32F769I_DISCOVERY_BSP_VERSION_MAIN</code></td>
<td>0x02</td>
</tr>
<tr>
<td>STM32F769I Discovery BSP Driver version number V2.0.0.</td>
<td></td>
</tr>
<tr>
<td><code>__STM32F769I_DISCOVERY_BSP_VERSION_SUB1</code></td>
<td>0x00</td>
</tr>
<tr>
<td><code>__STM32F769I_DISCOVERY_BSP_VERSION_SUB2</code></td>
<td>0x00</td>
</tr>
<tr>
<td><code>__STM32F769I_DISCOVERY_BSP_VERSION_RC</code></td>
<td>0x00</td>
</tr>
<tr>
<td><code>__STM32F769I_DISCOVERY_BSP_VERSION</code></td>
<td></td>
</tr>
</tbody>
</table>
Define Documentation

#define __STM32F769I_DISCOVERY_BSP_VERSION

Value:

\(((\text{__STM32F769I\_DISCOVERY\_BSP\_VERSION\_MAIN} \ll 24)\)\|
\((\text{__STM32F769I\_DISCOVERY\_BSP\_VERSION\_SUB1} \ll 16)\)\|
\((\text{__STM32F769I\_DISCOVERY\_BSP\_VERSION\_SUB2} \ll 8 )\)\|
\((\text{__STM32F769I\_DISCOVERY\_BSP\_VERSION\_RC})\))

Definition at line 73 of file stm32f769i_discovery.c.

Referenced by BSP_GetVersion().

#define __STM32F769I_DISCOVERY_BSP_VERSION_MAIN (0x02)

STM32F769I Discovery BSP Driver version number V2.0.0.

[31:24] main version

Definition at line 69 of file stm32f769i_discovery.c.

#define __STM32F769I_DISCOVERY_BSP_VERSION_RC (0x00)

[7:0] release candidate

Definition at line 72 of file stm32f769i_discovery.c.

#define __STM32F769I_DISCOVERY_BSP_VERSION_SUB1 (0x00)
[23:16] sub1 version

Definition at line 70 of file stm32f769i_discovery.c.

#define __STM32F769I_DISCOVERY_BSP_VERSION_SUB2 (0x00)

[15:8] sub2 version

Definition at line 71 of file stm32f769i_discovery.c.

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery
BSP User Manual by doxygen 1.7.6.1
## LCD Private Macros

```
STM32F769I_DISCOVERY LCD
```
**Defines**

```c
#define ABS(X) ((X) > 0 ? (X) : -(X))
#define POLY_X(Z) ((int32_t)((Points + (Z))->X))
#define POLY_Y(Z) ((int32_t)((Points + (Z))->Y))
```
Define Documentation

#define ABS ( X ) ((X) > 0 ? (X) : -(X))
Definition at line 167 of file stm32f769i_discovery_lcd.c.
Referenced by BSP_LCD_DrawLine(), and FillTriangle().

#define POLY_X ( Z ) ((int32_t)((Points + (Z))->X))
Definition at line 169 of file stm32f769i_discovery_lcd.c.
Referenced by BSP_LCD_FillPolygon().

#define POLY_Y ( Z ) ((int32_t)((Points + (Z))->Y))
Definition at line 170 of file stm32f769i_discovery_lcd.c.
Referenced by BSP_LCD_FillPolygon().
STM32F769I-Discovery BSP User Manual

Directories

LCD Private Variables

STM32F769I_DISCOVERY_LCD
### Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static uint32_t</code></td>
<td><strong>ActiveLayer</strong></td>
<td>LTDC_ACTIVE_LAYER_BACKGROUND Default Active LTDC Layer in which drawing is made is LTDC Layer Background.</td>
</tr>
<tr>
<td><code>static LCD_DrawPropTypeDef</code></td>
<td><strong>DrawProp</strong> [LTDC_MAX_LAYER_NUMBER]</td>
<td>Current Drawing Layer properties variable.</td>
</tr>
</tbody>
</table>
**Variable Documentation**

`uint32_t ActiveLayer = LTDC_ACTIVE_LAYER_BACKGROUND` [static]

Default Active LTDC Layer in which drawing is made is LTDC Layer Background.

Definition at line 227 of file `stm32f769i_discovery_lcd.c`.


**LCD_DrawPropTypeDef DrawProp[LTDC_MAX_LAYER_NUMBER]**

Current Drawing Layer properties variable.

Definition at line 232 of file `stm32f769i_discovery_lcd.c`.

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery
BSP User Manual by [doxygen] 1.7.6.1
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**STM32F769I_DISCOVERY_AUDIO Private Variables**

**STM32F769I_DISCOVERY_AUDIO**
### Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUDIO_DrvTypeDef *</td>
<td>audio_drv</td>
</tr>
<tr>
<td>SAI_HandleTypeDef</td>
<td>haudio_out_sai</td>
</tr>
<tr>
<td>SAI_HandleTypeDef</td>
<td>haudio_in_sai</td>
</tr>
<tr>
<td>AUDIOIN_TypeDef</td>
<td>hAudioIn</td>
</tr>
<tr>
<td>DFSDM_Channel_HandleTypeDef</td>
<td>hAudioInTopLeftChannel</td>
</tr>
<tr>
<td>DFSDM_Channel_HandleTypeDef</td>
<td>hAudioInTopRightChannel</td>
</tr>
<tr>
<td>DFSDM_Filter_HandleTypeDef</td>
<td>hAudioInTopLeftFilter</td>
</tr>
<tr>
<td>DFSDM_Filter_HandleTypeDef</td>
<td>hAudioInTopRightFilter</td>
</tr>
<tr>
<td>DMA_HandleTypeDef</td>
<td>hDmaTopLeft</td>
</tr>
<tr>
<td>DMA_HandleTypeDef</td>
<td>hDmaTopRight</td>
</tr>
<tr>
<td>DFSDM_Channel_HandleTypeDef</td>
<td>hAudioInBottomLeftChannel</td>
</tr>
<tr>
<td>DFSDM_Channel_HandleTypeDef</td>
<td>hAudioInBottomRightChannel</td>
</tr>
<tr>
<td>DFSDM_Filter_HandleTypeDef</td>
<td>hAudioInBottomLeftFilter</td>
</tr>
<tr>
<td>DFSDM_Filter_HandleTypeDef</td>
<td>hAudioInBottomRightFilter</td>
</tr>
<tr>
<td>DMA_HandleTypeDef</td>
<td>hDmaBottomLeft</td>
</tr>
<tr>
<td>DMA_HandleTypeDef</td>
<td>hDmaBottomRight</td>
</tr>
<tr>
<td>static int32_t *</td>
<td>pScratchBuff [2 * DEFAULT_AUDIO_IN_CHANNEL]</td>
</tr>
<tr>
<td>static __IO int32_t</td>
<td>ScratchSize</td>
</tr>
<tr>
<td>static uint8_t</td>
<td>AudioIn_ChannelNumber =</td>
</tr>
<tr>
<td>static uint16_t</td>
<td>DEFAULT_AUDIO_IN_CHANNEL_</td>
</tr>
<tr>
<td>static uint32_t</td>
<td>AudioIn_Device = INPUT_DEVICE_DIGITAL_MIC</td>
</tr>
<tr>
<td>static uint32_t</td>
<td>DmaTopLeftRecHalfCplt = 0</td>
</tr>
<tr>
<td>static uint32_t</td>
<td>DmaTopLeftRecCplt = 0</td>
</tr>
<tr>
<td>static uint32_t</td>
<td>DmaTopRightRecHalfCplt = 0</td>
</tr>
<tr>
<td>static uint32_t</td>
<td>DmaTopRightRecCplt = 0</td>
</tr>
<tr>
<td>static uint32_t</td>
<td>DmaBottomLeftRecHalfCplt = 0</td>
</tr>
<tr>
<td>static uint32_t</td>
<td>DmaBottomLeftRecCplt = 0</td>
</tr>
<tr>
<td>static uint32_t</td>
<td>DmaBottomRightRecHalfCplt = 0</td>
</tr>
<tr>
<td>static uint32_t</td>
<td>DmaBottomRightRecCplt = 0</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-----------------</td>
</tr>
<tr>
<td>static</td>
<td>__IO uint32_t</td>
</tr>
<tr>
<td>AppBuffTrigger</td>
<td>= 0</td>
</tr>
<tr>
<td>static</td>
<td>__IO uint32_t</td>
</tr>
<tr>
<td>AppBuffHalf</td>
<td>= 0</td>
</tr>
</tbody>
</table>
### Variable Documentation

<table>
<thead>
<tr>
<th>Variable</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>__IO uint32_t <strong>AppBuffHalf</strong> = 0 [static]</td>
<td>Definition at line 257 of file stm32f769i_discovery_audio.c. Referenced by <strong>BSP_AUDIO_IN_Record()</strong>, <strong>BSP_AUDIO_IN_Stop()</strong>, <strong>HAL_DFSDM_FilterRegConvCpltCallback()</strong>, and <strong>HAL_DFSDM_FilterRegConvHalfCpltCallback()</strong>.</td>
</tr>
<tr>
<td>__IO uint32_t <strong>AppBuffTrigger</strong> = 0 [static]</td>
<td>Definition at line 256 of file stm32f769i_discovery_audio.c. Referenced by <strong>BSP_AUDIO_IN_Record()</strong>, <strong>BSP_AUDIO_IN_Stop()</strong>, <strong>HAL_DFSDM_FilterRegConvCpltCallback()</strong>, and <strong>HAL_DFSDM_FilterRegConvHalfCpltCallback()</strong>.</td>
</tr>
<tr>
<td>AUDIO_DrvTypeDef* <strong>audio_drv</strong></td>
<td>Definition at line 216 of file stm32f769i_discovery_audio.c. Referenced by <strong>BSP_AUDIO_IN_InitEx()</strong>, <strong>BSP_AUDIO_IN_Stop()</strong>, <strong>BSP_AUDIO_OUT_Init()</strong>, <strong>BSP_AUDIO_OUT_Pause()</strong>, <strong>BSP_AUDIO_OUT_Play()</strong>, <strong>BSP_AUDIO_OUT_Resume()</strong>, <strong>BSP_AUDIO_OUT_SetMute()</strong>, <strong>BSP_AUDIO_OUT_SetOutputMode()</strong>, <strong>BSP_AUDIO_OUT_SetVolume()</strong>, and <strong>BSP_AUDIO_OUT_Stop()</strong>.</td>
</tr>
<tr>
<td>uint8_t <strong>AudioIn_ChannelNumber</strong> = DEFAULT_AUDIO_IN_CHANNEL</td>
<td>Definition at line 241 of file stm32f769i_discovery_audio.c.</td>
</tr>
</tbody>
</table>
Referenced by `BSP_AUDIO_IN_AllocScratch()`, `BSP_AUDIO_IN_GetChannelNumber()`, `BSP_AUDIO_IN_InitEx()`, `BSP_AUDIO_INPause()`, `BSP_AUDIO_IN_Record()`, `BSP_AUDIO_IN_Resume()`, `BSP_AUDIO_IN_Stop()`, `DFSDMx_ChannelMspDeInit()`, `DFSDMx_ChannelMspInit()`, `DFSDMx_DeInit()`, `DFSDMx_FilterMspDeInit()`, `DFSDMx_FilterMspInit()`, `DFSDMx_Init()`, `HAL_DFSDM_FilterRegConvCpltCallback()`, and `HAL_DFSDM_FilterRegConvHalfCpltCallback()`.

```c
uint16_t AudioIn_Device = INPUT_DEVICE_DIGITAL_MIC [static]
```

Definition at line 243 of file `stm32f769i_discovery_audio.c`.

Referenced by `BSP_AUDIO_IN_DeInit()`, `BSP_AUDIO_IN_InitEx()`, `BSP_AUDIO_IN_MspDeInit()`, `BSP_AUDIO_IN_MspInit()`, `BSP_AUDIO_IN_Record()`, and `BSP_AUDIO_IN_Stop()`.

```c
uint32_t DmaButtomLeftRecCplt = 0 [static]
```

Definition at line 251 of file `stm32f769i_discovery_audio.c`.

Referenced by `HAL_DFSDM_FilterRegConvCpltCallback()`.

```c
uint32_t DmaButtomLeftRecHalfCplt = 0 [static]
```

Definition at line 250 of file `stm32f769i_discovery_audio.c`.

Referenced by `HAL_DFSDM_FilterRegConvHalfCpltCallback()`.

```c
uint32_t DmaButtomRightRecCplt = 0 [static]
```

Definition at line 253 of file `stm32f769i_discovery_audio.c`.
Referenced by `HAL_DFSDM_FilterRegConvCpltCallback()`.

```c
uint32_t DmaBottomRightRecHalfCplt = 0 [static]
```
Definition at line 252 of file `stm32f769i_discovery_audio.c`.
Referenced by `HAL_DFSDM_FilterRegConvHalfCpltCallback()`.

```c
uint32_t DmaTopLeftRecCplt = 0 [static]
```
Definition at line 247 of file `stm32f769i_discovery_audio.c`.
Referenced by `HAL_DFSDM_FilterRegConvCpltCallback()`.

```c
uint32_t DmaTopLeftRecHalfCplt = 0 [static]
```
Definition at line 246 of file `stm32f769i_discovery_audio.c`.
Referenced by `HAL_DFSDM_FilterRegConvHalfCpltCallback()`.

```c
uint32_t DmaTopRightRecCplt = 0 [static]
```
Definition at line 249 of file `stm32f769i_discovery_audio.c`.
Referenced by `HAL_DFSDM_FilterRegConvCpltCallback()`.

```c
uint32_t DmaTopRightRecHalfCplt = 0 [static]
```
Definition at line 248 of file `stm32f769i_discovery_audio.c`.
Referenced by `HAL_DFSDM_FilterRegConvHalfCpltCallback()`.
SAI_HandleTypeDef haudio_in_sai

Definition at line 218 of file stm32f769i_discovery_audio.c.

Referenced by BSP_AUDIO_IN_InitEx(),
BSP_AUDIO_IN_MspDeInit(), BSP_AUDIO_IN_MspInit(),
BSP_AUDIO_IN_Record(), BSP_AUDIO_IN_Stop(),
SAIx_In_DeInit(), and SAIx_In_Init().

SAI_HandleTypeDef haudio_out_sai

Definition at line 217 of file stm32f769i_discovery_audio.c.

Referenced by BSP_AUDIO_OUT_ChangeBuffer(),
BSP_AUDIO_OUT_DeInit(), BSP_AUDIO_OUT_Init(),
BSP_AUDIO_OUT_PAUSE(), BSP_AUDIO_OUT_Play(),
BSP_AUDIO_OUT_Resume(),
BSP_AUDIO_OUT_setAudioFrameSlot(),
BSP_AUDIO_OUT_setFrequency(), BSP_AUDIO_OUT_Stop(),
SAIx_In_DeInit(), SAIx_In_Init(), SAIx_Out_DeInit(), and
SAIx_Out_Init().

AUDIOIN_TypeDef hAudioIn

Definition at line 221 of file stm32f769i_discovery_audio.c.

DFSDM_Channel_HandleTypeDef hAudioInButtomLeftChannel

Definition at line 230 of file stm32f769i_discovery_audio.c.

Referenced by DFSDMx_DeInit(), and DFSDMx_Init().

DFSDM_Filter_HandleTypeDef hAudioInButtomLeftFilter
Definition at line 232 of file stm32f769i_discovery_audio.c.

Referenced by BSP_AUDIO_IN_Pause(),
BSP_AUDIO_IN_Record(), BSP_AUDIO_IN_Resume(),
BSP_AUDIO_IN_Stop(), DFSDMx_DeInit(),
DFSDMx_FilterMspInit(), DFSDMx_Init(),
HAL_DFSDM_FilterRegConvCpltCallback(), and
HAL_DFSDM_FilterRegConvHalfCpltCallback().

DFSDM_Channel_HandleTypeDef hAudioInBottomRightChannel

Definition at line 231 of file stm32f769i_discovery_audio.c.

Referenced by DFSDMx_DeInit(), and DFSDMx_Init().

DFSDM_Filter_HandleTypeDef hAudioInBottomRightFilter

Definition at line 233 of file stm32f769i_discovery_audio.c.

Referenced by BSP_AUDIO_IN_Pause(),
BSP_AUDIO_IN_Record(), BSP_AUDIO_IN_Resume(),
BSP_AUDIO_IN_Stop(), DFSDMx_DeInit(),
DFSDMx_FilterMspInit(), and DFSDMx_Init().

DFSDM_Channel_HandleTypeDef hAudioInTopLeftChannel

Definition at line 223 of file stm32f769i_discovery_audio.c.

Referenced by DFSDMx_DeInit(), and DFSDMx_Init().

DFSDM_Filter_HandleTypeDef hAudioInTopLeftFilter

Definition at line 225 of file stm32f769i_discovery_audio.c.
Referenced by `BSP_AUDIO_IN_InitEx()`, `BSP_AUDIO_INPause()`, `BSP_AUDIO_IN_Record()`, `BSP_AUDIO_IN_Resume()`, `BSP_AUDIO_IN_Stop()`, `DFSDMx_DeInit()`, `DFSDMx_FilterMspInit()`, `DFSDMx_Init()`, `HAL_DFSDM_FilterRegConvCpltCallback()`, and `HAL_DFSDM_FilterRegConvHalfCpltCallback()`.

### `DFSDM_Channel_HandleTypeDef hAudioInTopRightChannel`

Definition at line 224 of file `stm32f769i_discovery_audio.c`. Referenced by `DFSDMx_DeInit()`, and `DFSDMx_Init()`.

### `DFSDM_Filter_HandleTypeDef hAudioInTopRightFilter`

Definition at line 226 of file `stm32f769i_discovery_audio.c`. Referenced by `BSP_AUDIO_IN_Pause()`, `BSP_AUDIO_IN_Record()`, `BSP_AUDIO_IN_Resume()`, `BSP_AUDIO_IN_Stop()`, `DFSDMx_DeInit()`, `DFSDMx_FilterMspInit()`, `DFSDMx_Init()`, `HAL_DFSDM_FilterRegConvCpltCallback()`, and `HAL_DFSDM_FilterRegConvHalfCpltCallback()`.

### `DMA_HandleTypeDef hDmaButtomLeft`

Definition at line 234 of file `stm32f769i_discovery_audio.c`. Referenced by `DFSDMx_FilterMspDeInit()`, and `DFSDMx_FilterMspInit()`.

### `DMA_HandleTypeDef hDmaButtomRight`
Definition at line 235 of file stm32f769i_discovery_audio.c.

Referenced by DFSDMx_FilterMspDeInit(), and DFSDMx_FilterMspInit().

**DMA_HandleTypeDef hDmaTopLeft**

Definition at line 227 of file stm32f769i_discovery_audio.c.

Referenced by DFSDMx_FilterMspDeInit(), and DFSDMx_FilterMspInit().

**DMA_HandleTypeDef hDmaTopRight**

Definition at line 228 of file stm32f769i_discovery_audio.c.

Referenced by DFSDMx_FilterMspDeInit(), and DFSDMx_FilterMspInit().

`int32_t* pScratchBuff[2 *DEFAULT_AUDIO_IN_CHANNEL_NBR] [static]`

Definition at line 238 of file stm32f769i_discovery_audio.c.

Referenced by BSP_AUDIO_IN_AllocScratch(), BSP_AUDIO_IN_Record(), BSP_AUDIO_IN_Resume(), HAL_DFSDM_FilterRegConvCpltCallback(), and HAL_DFSDM_FilterRegConvHalfCpltCallback().

`__IO int32_t ScratchSize [static]`

Definition at line 239 of file stm32f769i_discovery_audio.c.

Referenced by BSP_AUDIO_IN_AllocScratch(),
BSP_AUDIO_IN_Record(), BSP_AUDIO_IN_Resume(), HAL_DFSDM_FilterRegConvCpltCallback(), and HAL_DFSDM_FilterRegConvHalfCpltCallback.
# STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
</table>

## Directories

**STM32F769I_DISCOVERY_AUDIO**

**Exported Constants**

**STM32F769I_DISCOVERY AUDIO**
<table>
<thead>
<tr>
<th>Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSP Audio Sample Rate</td>
</tr>
</tbody>
</table>
Defines

```c
#define CODEC_AUDIOFRAME_SLOT_0123  SAI_SLOTACTIVE_0 | SAI_SLOTACTIVE_1 | SAI_SLOTACTIVE_2 | SAI_SLOTACTIVE_3
CODEC_AudioFrame_SLOT_TDMMode In W8994 codec the AudioFrame SLOT_TDMMode is:
| CODEC_SLOT0 Left | CODEC_SLOT1 Left | CODEC_SLOT0 Right | CODEC_SLOT1 Right |
+------------------+------------------+--------------------+-------------------+
```

```c
#define AUDIO_OUT_SAIx_CLK_ENABLE() __HAL_RCC_SAI1_CLK_ENABLE()
#define AUDIO_OUT_SAIx_CLK_DISABLE() __HAL_RCC_SAI1_CLK_DISABLE()
#define AUDIO_OUT_SAIx_AF GPIO_AF6_SAI1
#define AUDIO_OUT_SAIx_MCLK_ENABLE() __HAL_RCC_GPIOG_CLK_ENABLE()
#define AUDIO_OUT_SAIx_MCLK_GPIO_PORT GPIOG
#define AUDIO_OUT_SAIx_MCLK_PIN GPIO_PIN_7
#define AUDIO_OUT_SAIx_SD_FS_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define AUDIO_OUT_SAIx_SD_FS_SCK_GPIO_PORT GPIOE
#define AUDIO_OUT_SAIx_FS_PIN GPIO_PIN_4
#define AUDIO_OUT_SAIx_SCK_PIN GPIO_PIN_5
#define AUDIO_OUT_SAIx_SD_PIN GPIO_PIN_6
#define AUDIO_OUT_SAIx_DMAx_CLK_ENABLE() __HAL_RCC_DMA2_CLK_ENABLE()
#define AUDIO_OUT_SAIx_DMAx_STREAM DMA2_Stream1
#define AUDIO_OUT_SAIx_DMAx_CHANNEL DMA_CHANNEL_0
#define AUDIO_OUT_SAIx_DMAx_IRQ DMA2_Stream1_IRQn
#define AUDIO_OUT_SAIx_DMAx_PERIPH_DATA_SIZE DMA_PDATAALIGN_HALFWORD
#define AUDIO_OUT_SAIx_DMAx_MEM_DATA_SIZE DMA_MDATAALIGN_HALFWORD
#define DMA_MAX_SZE 0xFFFF
#define AUDIO_OUT_SAIx_DMAx_IRQHandler DMA2_Stream1_IRQHandler
#define AUDIO_IN_SAIx_CLK_ENABLE() __HAL_RCC_SAI1_CLK_ENABLE()
#define AUDIO_IN_SAIx_CLK_DISABLE() __HAL_RCC_SAI1_CLK_DISABLE()
```
#define AUDIO_IN_SAIx_CLK_DISABLE() __HAL_RCC_SAIx_CLK_DISABLE()
#define AUDIO_IN_SAIx_AF GPIO_AF6_SAIx
#define AUDIO_IN_SAIx_SD_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define AUDIO_IN_SAIx_SD_GPIO_PORT GPIOE
#define AUDIO_IN_SAIx_SD_PIN GPIO_PIN_3
#define AUDIO_IN_SAIx_DMAx_CLK_ENABLE() __HAL_RCC_DMAx_CLK_ENABLE()
#define AUDIO_IN_SAIx_DMAx_STREAM DMAx_Stream
#define AUDIO_IN_SAIx_DMAx_CHANNEL DMA_CHANNEL
#define AUDIO_IN_SAIx_DMAx_IRQ DMAx_IRQHandler
#define AUDIO_IN_SAIx_DMAx_PERIPH_DATA_SIZE DMA_PERIPH_DATA_SIZE
#define AUDIO_IN_SAIx_DMAx_MEM_DATA_SIZE DMA_MEM_DATA_SIZE
#define AUDIO_IN_INT_GPIO_ENABLE() __HAL_RCC_GPIOJ_CLK_ENABLE()
#define AUDIO_IN_INT_GPIO_PORT GPIOJ
#define AUDIO_IN_INT_GPIO_PIN GPIO_PIN_12
#define AUDIO_IN_INT_IRQ EXTI15_10_IRQHandler
#define AUDIO_DFSDMx_TOP_RIGHT_CHANNEL DFSDM_CHANNEL
#define AUDIO_DFSDMx_TOP_LEFT_CHANNEL DFSDM_CHANNEL
#define AUDIO_DFSDMx_BUTTOM_RIGHT_CHANNEL DFSDM_CHANNEL
#define AUDIO_DFSDMx_BUTTOM_LEFT_CHANNEL DFSDM_CHANNEL
#define AUDIO_DFSDMx_TOP_LEFT_FILTER DFSDM1_Filter
#define AUDIO_DFSDMx_TOP_RIGHT_FILTER DFSDM1_Filter
#define AUDIO_DFSDMx_CLK_ENABLE() __HAL_RCC_DFSDM1_CLK_ENABLE()
#define AUDIO_DFSDMx_CKOUT_PIN GPIO_PIN_3
#define AUDIO_DFSDMx_CKOUT_DMIC_GPIO_PORT GPIOJ
#define AUDIO_DFSDMx_CKOUT_DMIC_GPIO_CLK_ENABLE() __HAL_RCC_GPIOJ_CLK_ENABLE()
#define AUDIO_DFSDMx_DMIC_DATIN1_PIN GPIO_PIN_3
#define AUDIO_DFSDMx_DMIC_DATIN5_PIN GPIO_PIN_11
#define AUDIO_DFSDMx_DMIC_DATIN_GPIO_PORT GPIOJ
#define AUDIO_DFSDMx_DMIC_DATIN_GPIO_CLK_ENABLE() __HAL_RCC_GPIOJ_CLK_ENABLE()
#define AUDIO_DFSDMx_DMIC_DATIN_AF GPIO_AF3_DFSDM1
#define AUDIO_DFSDMx_CKOUT_AF GPIO_AF3_DFSDM1
#define AUDIO_DFSDMx_DMAx_CLK_ENABLE() __HAL_RCC_DMAx_CLK_ENABLE()
#define AUDIO_DFSDMx_DMAx_CHANNEL DMA_CHANNEL_8
#define AUDIO_DFSDMx_DMAx_PERIPH_DATA_SIZE DMA_PDATAALIGN_WORD
#define AUDIO_DFSDMx_DMAx_MEM_DATA_SIZE DMA_MDATAALIGN_WORD
#define AUDIO_DFSDMx_DMAx_TOP_LEFT_STREAM DMA2_Stream0
#define AUDIO_DFSDMx_DMAx_TOP_LEFT_IRQ DMA2_Stream0_IRQn
#define AUDIO_DFSDMx_DMAx_TOP_LEFT_IRQHandler DMA2_Stream0_IRQHandler
#define AUDIO_DFSDMx_DMAx_TOP_RIGHT_STREAM DMA2_Stream5
#define AUDIO_DFSDMx_DMAx_TOP_RIGHT_IRQ DMA2_Stream5_IRQn
#define AUDIO_DFSDMx_DMAx_TOP_RIGHT_IRQHandler DMA2_Stream5_IRQHandler
#define AUDIO_DFSDMx_DMAx_BUTTOM_LEFT_STREAM DMA2_Stream6
#define AUDIO_DFSDMx_DMAx_BUTTOM_LEFT_IRQ DMA2_Stream6_IRQn
#define AUDIO_DFSDMx_DMAx_BUTTOM_LEFT_IRQHandler DMA2_Stream6_IRQHandler
#define AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_STREAM DMA2_Stream7
#define AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_IRQ DMA2_Stream7_IRQn
#define AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_IRQHandler DMA2_Stream7_IRQHandler
#define AUDIO_IN_IRQ_PREPRIO ((uint32_t)0x0F)
#define AUDIODATA_SIZE 2 /* 16-bits audio data size */
#define AUDIO_OK ((uint8_t)0)
#define AUDIO_ERROR ((uint8_t)1)
#define AUDIO_TIMEOUT ((uint8_t)2)
#define DEFAULT_AUDIO_IN_FREQ BSP_AUDIO_FREQUENCY_16K
#define DEFAULT_AUDIO_IN_BIT_RESOLUTION ((uint8_t)16)
#define DEFAULT_AUDIO_IN_CHANNEL_NBR ((uint8_t)2)
#define DEFAULT_AUDIO_IN_VOLUME ((uint16_t)64)
#define OUTPUT_DEVICE_HEADPHONE1 OUTPUT_DEVICE_HEADPHONE
#define OUTPUT_DEVICE_HEADPHONE2 OUTPUT_DEVICE_SPEAKER /* Headphone2 is connected to Speaker output of the wm8994 */
#define INPUT_DEVICE_DIGITAL_MIC ((uint16_t)0)
#define INPUT_DEVICE_ANALOG_MIC INPUT_DEVICE_INPUT_LINE_1
Define Documentation

<table>
<thead>
<tr>
<th>Define</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#define AUDIO_DFSDMx_BUTTOM_LEFT_CHANNEL</code></td>
<td>DFSDM_CHANNEL_5</td>
</tr>
<tr>
<td>Definition at line 167 of file <code>stm32f769i_discovery_audio.h</code>.</td>
<td>Referenced by <code>DFSDMx_Init()</code>.</td>
</tr>
<tr>
<td><code>#define AUDIO_DFSDMx_BUTTOM_LEFT_FILTER</code></td>
<td>DFSDM1_Filter2</td>
</tr>
<tr>
<td>Definition at line 171 of file <code>stm32f769i_discovery_audio.h</code>.</td>
<td>Referenced by <code>DFSDMx_Init()</code>.</td>
</tr>
<tr>
<td><code>#define AUDIO_DFSDMx_BUTTOM_RIGHT_CHANNEL</code></td>
<td>DFSDM_CHANNEL_4</td>
</tr>
<tr>
<td>Definition at line 166 of file <code>stm32f769i_discovery_audio.h</code>.</td>
<td>Referenced by <code>DFSDMx_Init()</code>.</td>
</tr>
<tr>
<td><code>#define AUDIO_DFSDMx_BUTTOM_RIGHT_FILTER</code></td>
<td>DFSDM1_Filter3</td>
</tr>
<tr>
<td>Definition at line 172 of file <code>stm32f769i_discovery_audio.h</code>.</td>
<td>Referenced by <code>DFSDMx_Init()</code>.</td>
</tr>
<tr>
<td><code>#define AUDIO_DFSDMx_CKOUT_AF</code></td>
<td>GPIO_AF3_DFSDM1</td>
</tr>
<tr>
<td>Definition at line 183 of file <code>stm32f769i_discovery_audio.h</code>.</td>
<td>Referenced by <code>DFSDMx_ChannelMspInit()</code>.</td>
</tr>
</tbody>
</table>
```plaintext
#define AUDIO_DFSDMx_CKOUT_DMIC_GPIO_CLK_ENABLE () __HAL_RCC_GPIOD_CLK_ENABLE()

Definition at line 177 of file stm32f769i_discovery_audio.h.
Referenced by DFSDMx_ChannelMspInit().

#define AUDIO_DFSDMx_CKOUT_DMIC_GPIO_PORT GPIOD

Definition at line 176 of file stm32f769i_discovery_audio.h.
Referenced by DFSDMx_ChannelMspDeInit(), and DFSDMx_ChannelMspInit().

#define AUDIO_DFSDMx_CKOUT_PIN GPIO_PIN_3

Definition at line 175 of file stm32f769i_discovery_audio.h.
Referenced by DFSDMx_ChannelMspDeInit(), and DFSDMx_ChannelMspInit().

#define AUDIO_DFSDMx_CLK_ENABLE () __HAL_RCC_DFSDM1

Definition at line 174 of file stm32f769i_discovery_audio.h.
Referenced by DFSDMx_ChannelMspInit(), and DFSDMx_FilterMspInit().

#define AUDIO_DFSDMx_DMAx_BUTTOM_LEFT_IRQ DMA2_Stream6_IRQn

Definition at line 200 of file stm32f769i_discovery_audio.h.
Referenced by DFSDMx_FilterMspInit().
```
#define AUDIO_DFSDMx_DMAx_BUTTOM_LEFT_IRQHandler DMA2_Stream6_IRQHandler
Definition at line 201 of file stm32f769i_discovery_audio.h.

#define AUDIO_DFSDMx_DMAx_BUTTOM_LEFT_STREAM DMA2_Stream6
Definition at line 199 of file stm32f769i_discovery_audio.h.
Referenced by DFSDMx_FilterMspInit().

#define AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_IRQ DMA2_Stream7_IRQn
Definition at line 204 of file stm32f769i_discovery_audio.h.
Referenced by DFSDMx_FilterMspInit().

#define AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_IRQHandler DMA2_Stream7_IRQHandler
Definition at line 205 of file stm32f769i_discovery_audio.h.

#define AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_STREAM DMA2_Stream7
Definition at line 203 of file stm32f769i_discovery_audio.h.
Referenced by DFSDMx_FilterMspInit().

#define AUDIO_DFSDMx_DMAx_CHANNEL DMA_CHANNEL_8
Definition at line 187 of file stm32f769i_discovery_audio.h.
Referenced by DFSDMx_FilterMspInit().
#define AUDIO_DFSDMx_DMAx_CLK_ENABLE () __HAL_RCC_DMA2_CLK_ENABLE()

Definition at line 186 of file stm32f769i_discovery_audio.h.
Referenced by DFSDMx_FilterMspInit().

#define AUDIO_DFSDMx_DMAx_MEM_DATA_SIZE DMA_MDATAA

Definition at line 189 of file stm32f769i_discovery_audio.h.
Referenced by DFSDMx_FilterMspInit().

#define AUDIO_DFSDMx_DMAx_PERIPH_DATA_SIZE DMA_PDAT.

Definition at line 188 of file stm32f769i_discovery_audio.h.
Referenced by DFSDMx_FilterMspInit().

#define AUDIO_DFSDMx_DMAx_TOP_LEFT_IRQ DMA2_Stream0

Definition at line 192 of file stm32f769i_discovery_audio.h.
Referenced by DFSDMx_FilterMspInit().

#define AUDIO_DFSDMx_DMAx_TOP_LEFT_IRQHandler DMA2_Stream0_IRQHandler

Definition at line 193 of file stm32f769i_discovery_audio.h.

#define AUDIO_DFSDMx_DMAx_TOP_LEFT_STREAM DMA2_Stream

Definition at line 191 of file stm32f769i_discovery_audio.h.
Referenced by `DFSDMx_FilterMspInit()`.

```c
#define AUDIO_DFSDMx_DMAx_TOP_RIGHT_IRQ DMA2_Stream5_IRQn
```
Definition at line 196 of file `stm32f769i_discovery_audio.h`.
Referenced by `DFSDMx_FilterMspInit()`.

```c
#define AUDIO_DFSDMx_DMAx_TOP_RIGHT_IRQHandler DMA2_Stream5_IRQHandler
```
Definition at line 197 of file `stm32f769i_discovery_audio.h`.

```c
#define AUDIO_DFSDMx_DMAx_TOP_RIGHT_STREAM DMA2_Stream5
```
Definition at line 195 of file `stm32f769i_discovery_audio.h`.
Referenced by `DFSDMx_FilterMspInit()`.

```c
#define AUDIO_DFSDMx_DMIC_DATIN1_PIN GPIO_PIN_3
```
Definition at line 178 of file `stm32f769i_discovery_audio.h`.
Referenced by `DFSDMx_ChannelMspDeInit()`, and `DFSDMx_ChannelMspInit()`.

```c
#define AUDIO_DFSDMx_DMIC_DATIN5_PIN GPIO_PIN_11
```
Definition at line 179 of file `stm32f769i_discovery_audio.h`.
Referenced by `DFSDMx_ChannelMspDeInit()`, and `DFSDMx_ChannelMspInit()`.


**#define AUDIO_DFSDMx_DMIC_DATIN_AF GPIO_AF3_DFSDM1**

Definition at line 182 of file stm32f769i_discovery_audio.h.

Referenced by DFSDMx_ChannelMspInit().

**#define AUDIO_DFSDMx_DMIC_DATIN_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOC_CLK_ENABLE()**

Definition at line 181 of file stm32f769i_discovery_audio.h.

Referenced by DFSDMx_ChannelMspInit().

**#define AUDIO_DFSDMx_DMIC_DATIN_GPIO_PORT GPIOC**

Definition at line 180 of file stm32f769i_discovery_audio.h.

Referenced by DFSDMx_ChannelMspDeInit(), and DFSDMx_ChannelMspInit().

**#define AUDIO_DFSDMx_TOP_LEFT_CHANNEL DFSDM_CHANNEL_1**

Definition at line 165 of file stm32f769i_discovery_audio.h.

Referenced by DFSDMx_Init().

**#define AUDIO_DFSDMx_TOP_LEFT_FILTER DFSDM1_Filter0**

Definition at line 169 of file stm32f769i_discovery_audio.h.

Referenced by DFSDMx_Init().

**#define AUDIO_DFSDMx_TOP_RIGHT_CHANNEL DFSDM_CHANNEL_0**
Definition at line 164 of file `stm32f769i_discovery_audio.h`.
Referenced by `DFSDMx_Init()`.

```c
#define AUDIO_DFSDMx_TOP_RIGHT_FILTER DFSDM1_Filter1
```
Definition at line 170 of file `stm32f769i_discovery_audio.h`.
Referenced by `DFSDMx_Init()`.

```c
#define AUDIO_ERROR ((uint8_t)1)
```
Definition at line 219 of file `stm32f769i_discovery_audio.h`.
Referenced by `BSP_AUDIO_IN_InitEx()`, `BSP_AUDIO_IN_Pause()`, `BSP_AUDIO_IN_Record()`, `BSP_AUDIO_IN_Resume()`, `BSP_AUDIO_IN_Stop()`, `BSP_AUDIO_OUT_Init()`, `BSP_AUDIO_OUT_Pause()`, `BSP_AUDIO_OUT_Play()`, `BSP_AUDIO_OUT_Resume()`, `BSP_AUDIO_OUT_SetMute()`, `BSP_AUDIO_OUT_SetOutputMode()`, `BSP_AUDIO_OUT_SetVolume()`, `BSP_AUDIO_OUT_Stop()`, `DFSDMx_DeInit()`, and `DFSDMx_Init()`.

```c
#define AUDIO_IN_INT_GPIO_ENABLE ( ) __HAL_RCC_GPIOJ_CLK_ENABLE()
```
Definition at line 158 of file `stm32f769i_discovery_audio.h`.
Referenced by `SAI_AUDIO_IN_MspInit()`.

```c
#define AUDIO_IN_INT_GPIO_PIN GPIO_PIN_12
```
Definition at line 160 of file `stm32f769i_discovery_audio.h`.
Referenced by `SAI_AUDIO_IN_MspDelInit()`, and
SAI_AUDIO_IN_MspInit().

#define AUDIO_IN_INT_GPIO_PORT GPIOJ

Definition at line 159 of file stm32f769i_discovery_audio.h.

Referenced by SAI_AUDIO_IN_MspDeInit(), and SAI_AUDIO_IN_MspInit().

#define AUDIO_IN_INT_IRQ EXTI15_10_IRQHandler

Definition at line 161 of file stm32f769i_discovery_audio.h.

Referenced by SAI_AUDIO_IN_MspInit().

#define AUDIO_IN_IRQ_PREPRIO ((uint32_t)0x0F)

Definition at line 208 of file stm32f769i_discovery_audio.h.

Referenced by SAI_AUDIO_IN_MspInit().

#define AUDIO_IN_SAIx SAI1_Block_B

Definition at line 142 of file stm32f769i_discovery_audio.h.

Referenced by BSP_AUDIO_IN_InitEx(), SAI_AUDIO_IN_MspDeInit(), SAI_AUDIO_IN_MspInit(), SAIx_In_DeInit(), and SAIx_In_Init().

#define AUDIO_IN_SAIx_AF GPIO_AF6_SAI1

Definition at line 145 of file stm32f769i_discovery_audio.h.
Referenced by `SAI_AUDIO_IN_MspInit`.

```c
#define AUDIO_IN_SAIx_CLK_DISABLE () __HAL_RCC_SAI1_CLK_DISABLE()
```
Definition at line 144 of file `stm32f769i_discovery_audio.h`.

Referenced by `SAI_AUDIO_IN_MspDeInit`.

```c
#define AUDIO_IN_SAIx_CLK_ENABLE () __HAL_RCC_SAI1_CLK_ENABLE()
```
Definition at line 143 of file `stm32f769i_discovery_audio.h`.

Referenced by `SAI_AUDIO_IN_MspInit`.

```c
#define AUDIO_IN_SAIx_DMAx_CHANNEL DMA_CHANNEL_1
```
Definition at line 153 of file `stm32f769i_discovery_audio.h`.

Referenced by `SAI_AUDIO_IN_MspInit`.

```c
#define AUDIO_IN_SAIx_DMAx_CLK_ENABLE () __HAL_RCC_DMA2_CLK_ENABLE()
```
Definition at line 151 of file `stm32f769i_discovery_audio.h`.

Referenced by `SAI_AUDIO_IN_MspInit`.

```c
#define AUDIO_IN_SAIx_DMAx_IRQ DMA2_Stream4_IRQHandler
```
Definition at line 154 of file `stm32f769i_discovery_audio.h`.

Referenced by `SAI_AUDIO_IN_MspDeInit`, and `SAI_AUDIO_IN_MspInit`.


```
#define AUDIO_IN_SAIx_DMAx_MEM_DATA_SIZE DMA_MDATAALIGN_HALFWORD
Definition at line 156 of file stm32f769i_discovery_audio.h.
Referenced by SAI_AUDIO_IN_MspInit().

#define AUDIO_IN_SAIx_DMAx_PERIPH_DATA_SIZE DMA_PDATAALIGN_HALFWORD
Definition at line 155 of file stm32f769i_discovery_audio.h.
Referenced by SAI_AUDIO_IN_MspInit().

#define AUDIO_IN_SAIx_DMAx_STREAM DMA2_Stream4
Definition at line 152 of file stm32f769i_discovery_audio.h.
Referenced by SAI_AUDIO_IN_MspInit().

#define AUDIO_IN_SAIx_SD_ENABLE () __HAL_RCC_GPIOE_CLK_ENABLE()
Definition at line 146 of file stm32f769i_discovery_audio.h.
Referenced by SAI_AUDIO_IN_MspInit().

#define AUDIO_IN_SAIx_SD_GPIO_PORT GPIOE
Definition at line 147 of file stm32f769i_discovery_audio.h.
Referenced by SAI_AUDIO_IN_MspDeInit(), and SAI_AUDIO_IN_MspInit().
```
```c
#define AUDIO_IN_SAIx_SD_PIN  GPIO_PIN_3

Definition at line 148 of file stm32f769i_discovery_audio.h.

Referenced by SAI_AUDIO_IN_MspDeInit(), and SAI_AUDIO_IN_MspInit().

#define AUDIO_OK  ((uint8_t)0)

Definition at line 218 of file stm32f769i_discovery_audio.h.

Referenced by BSP_AUDIO_IN_AllocScratch(),
BSP_AUDIO_IN_InitEx(), BSP_AUDIO_IN_Pause(),
BSP_AUDIO_IN_Record(), BSP_AUDIO_IN_Resume(),
BSP_AUDIO_IN_Stop(), BSP_AUDIO_OUT_Init(),
BSP_AUDIO_OUT_Pause(), BSP_AUDIO_OUT_Play(),
BSP_AUDIO_OUT_Resume(), BSP_AUDIO_OUT_SetMute(),
BSP_AUDIO_OUT_SetOutputMode(),
BSP_AUDIO_OUT_SetVolume(), BSP_AUDIO_OUT_Stop(),
DFSDMx_DeInit(), and DFSDMx_Init().

#define AUDIO_OUT_IRQ_PREPRIO  ((uint32_t)0x0E)

Definition at line 136 of file stm32f769i_discovery_audio.h.

Referenced by BSP_AUDIO_OUT_MspInit(), and DFSDMx_FilterMspInit().

#define AUDIO_OUT_SAIx  SAI1_Block_A

Definition at line 110 of file stm32f769i_discovery_audio.h.

Referenced by BSP_AUDIO_OUT_Init(),
BSP_AUDIO_OUT_MspDeInit(), BSP_AUDIO_OUT_MspInit(),
```
HAL_SAI_ErrorCallback(), SAIx_In_DeInit(), SAIx_In_Init(), SAIx_Out_DeInit(), and SAIx_Out_Init.

```c
#define AUDIO_OUT_SAIx_AF	GPIO_AF6_SAI1
```
Definition at line 113 of file stm32f769i_discovery_audio.h.
Referenced by BSP_AUDIO_OUT_MspInit().

```c
#define AUDIO_OUT_SAIx_CLK_DISABLE ( ) __HAL_RCC_SAI1_CLK_DISABLE()
```
Definition at line 112 of file stm32f769i_discovery_audio.h.
Referenced by BSP_AUDIO_OUT_MspDeInit().

```c
#define AUDIO_OUT_SAIx_CLK_ENABLE ( ) __HAL_RCC_SAI1_CLK_ENABLE()
```
Definition at line 111 of file stm32f769i_discovery_audio.h.
Referenced by BSP_AUDIO_OUT_MspInit().

```c
#define AUDIO_OUT_SAIx_DMAx_CHANNEL DMA_CHANNEL_0
```
Definition at line 127 of file stm32f769i_discovery_audio.h.
Referenced by BSP_AUDIO_OUT_MspInit().

```c
#define AUDIO_OUT_SAIx_DMAx_CLK_ENABLE ( ) __HAL_RCC_DMA2_CLK_ENABLE()
```
Definition at line 125 of file stm32f769i_discovery_audio.h.
Referenced by BSP_AUDIO_OUT_MspInit().
#define AUDIO_OUT_SAIx_DMAx_IRQ DMA2_Stream1_IRQHandler

Definition at line 128 of file stm32f769i_discovery_audio.h.

Referenced by BSP_AUDIO_OUT_MspDeInit(), and BSP_AUDIO_OUT_MspInit().

#define AUDIO_OUT_SAIx_DMAx_IRQHandler DMA2_Stream1_IRQHandler

Definition at line 133 of file stm32f769i_discovery_audio.h.

#define AUDIO_OUT_SAIx_DMAx_MEM_DATA_SIZE DMA_MDATAALIGN_HALFWORD

Definition at line 130 of file stm32f769i_discovery_audio.h.

Referenced by BSP_AUDIO_OUT_MspInit().

#define AUDIO_OUT_SAIx_DMAx_PERIPH_DATA_SIZE DMA_PDATAALIGN_HALFWORD

Definition at line 129 of file stm32f769i_discovery_audio.h.

Referenced by BSP_AUDIO_OUT_MspInit().

#define AUDIO_OUT_SAIx_DMAx_STREAM DMA2_Stream1

Definition at line 126 of file stm32f769i_discovery_audio.h.

Referenced by BSP_AUDIO_OUT_MspInit().

#define AUDIO_OUT_SAIx_FS_PIN GPIO_PIN_4
Definition at line 120 of file `stm32f769i_discovery_audio.h`.

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

```c
#define AUDIO_OUT_SAIx_MCLK_ENABLE () __HAL_RCC_GPIOG_CLK_ENABLE()
```

Definition at line 115 of file `stm32f769i_discovery_audio.h`.

Referenced by `BSP_AUDIO_OUT_MspInit()`.

```c
#define AUDIO_OUT_SAIx_MCLK_GPIO_PORT GPIOG
```

Definition at line 116 of file `stm32f769i_discovery_audio.h`.

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

```c
#define AUDIO_OUT_SAIx_MCLK_PIN GPIO_PIN_7
```

Definition at line 117 of file `stm32f769i_discovery_audio.h`.

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

```c
#define AUDIO_OUT_SAIx_SCK_PIN GPIO_PIN_5
```

Definition at line 121 of file `stm32f769i_discovery_audio.h`.

Referenced by `BSP_AUDIO_OUT_MspDeInit()`, and `BSP_AUDIO_OUT_MspInit()`.
```c
#define AUDIO_OUT_SAIx_SD_FS_CLK_ENABLE () __HAL_RCC_GPIOE_CLK_ENABLE()

Definition at line **118** of file `stm32f769i_discovery_audio.h`.

Referenced by `BSP_AUDIO_OUT_MspInit()`.

```}

```c
#define AUDIO_OUT_SAIx_SD_FS_SCK_GPIO_PORT GPIOE

Definition at line **119** of file `stm32f769i_discovery_audio.h`.

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

```}

```c
#define AUDIO_OUT_SAIx_SD_PIN GPIO_PIN_6

Definition at line **122** of file `stm32f769i_discovery_audio.h`.

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

```}

```c
#define AUDIO_TIMEOUT ((uint8_t)2)

Definition at line **220** of file `stm32f769i_discovery_audio.h`.

```}

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

Definition at line **215** of file `stm32f769i_discovery_audio.h`.

Referenced by `BSP_AUDIO_OUT_Play()`.

```}

```c
#define CODEC_AUDIOFRAME_SLOT_0123 SAI_SLOTACTIVE_0 | SAI_SLOTACTIVE_1 | SAI_SLOTACTIVE_2 | SAI_SLOTACTIVE_3

```
CODEC_AUDIOFRAME_SLOT_TDMMode In W8994 codec the Audio frame contains 4 slots: TDM Mode TDM format:

```
+------------------|-------
| CODEC_SLOT0 Left |
| CODEC_SLOT1 Left | CODEC_SLOT0 Right | CODEC_SLOT1 Right |
```

Definition at line 101 of file stm32f769i_discovery_audio.h.

Referenced by SAIx_Out_Init().

```c
#define CODEC_AUDIOFRAME_SLOT_02 SAI_SLOTACTIVE_0 | SAI_SLOTACTIVE_2
```

Definition at line 104 of file stm32f769i_discovery_audio.h.

Referenced by SAIx_In_Init().

```c
#define CODEC_AUDIOFRAME_SLOT_13 SAI_SLOTACTIVE_1 | SAI_SLOTACTIVE_3
```

Definition at line 106 of file stm32f769i_discovery_audio.h.

```c
#define DEFAULT_AUDIO_IN_BIT_RESOLUTION ((uint8_t)16)
```

Definition at line 224 of file stm32f769i_discovery_audio.h.

```c
#define DEFAULT_AUDIO_IN_CHANNEL_NBR ((uint8_t)2)
```

Definition at line 225 of file stm32f769i_discovery_audio.h.

```c
#define DEFAULT_AUDIO_IN_FREQ BSP_AUDIO_FREQUENCY_16K
```

Definition at line 223 of file stm32f769i_discovery_audio.h.
```c
#define DEFAULT_AUDIO_IN_VOLUME ((uint16_t)64)
Definition at line 226 of file stm32f769i_discovery_audio.h.

#define DMA_MAX_SZE 0xFFFF
Definition at line 131 of file stm32f769i_discovery_audio.h.

#define INPUT_DEVICE_ANALOG_MIC INPUTDEVICE_INPUT_PIN_1
Definition at line 241 of file stm32f769i_discovery_audio.h.

#define INPUT_DEVICE_DIGITAL_MIC ((uint16_t)0)
Definition at line 239 of file stm32f769i_discovery_audio.h.

Referenced by BSP_AUDIO_IN_DeInit(), BSP_AUDIO_IN_Init(),
BSP_AUDIO_IN_InitEx(), BSP_AUDIO_IN_MspDeInit(),
BSP_AUDIO_IN_MspInit(), BSP_AUDIO_IN_Record(), and
BSP_AUDIO_IN_Stop().

#define OUTPUT_DEVICE_HEADPHONE1 OUTPUT_DEVICE_HEADPHONE
Definition at line 232 of file stm32f769i_discovery_audio.h.

#define OUTPUT_DEVICE_HEADPHONE2 OUTPUT_DEVICE_SPEAKER /* Headphone2 is connected to Speaker output of the wm8994 */
Definition at line 233 of file stm32f769i_discovery_audio.h.
```
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**STM32F769I Discovery**  
**Low Level Exported Constants**

STM32F769I Discovery Low Level Exported Types
## Modules

<table>
<thead>
<tr>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F769I Discovery Low Level Led</td>
</tr>
<tr>
<td>STM32F769I Discovery Low Level Button</td>
</tr>
</tbody>
</table>
## Defines

<table>
<thead>
<tr>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#define OTG_HS_OVER_CURRENT_PIN GPIO_PIN_4</code></td>
<td>USB OTG HS Over Current signal.</td>
</tr>
<tr>
<td><code>#define OTG_HS_OVER_CURRENT_PORT GPIOD</code></td>
<td></td>
</tr>
<tr>
<td><code>#define OTG_HS_OVER_CURRENT_PORT_CLK_ENABLE()</code></td>
<td><code>__HAL_RCC_GPIOD_CLK_ENABLE()</code></td>
</tr>
<tr>
<td><code>#define SD_DETECT_PIN ((uint32_t)GPIO_PIN_15)</code></td>
<td>SD-detect signal.</td>
</tr>
<tr>
<td><code>#define SD_DETECT_GPIO_PORT ((GPIO_TypeDef*)GPIOI)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define SD_DETECT_GPIO_CLK_ENABLE()</code></td>
<td><code>__HAL_RCC_GPIOI_CLK_ENABLE()</code></td>
</tr>
<tr>
<td><code>#define SD_DETECT_GPIO_CLK_DISABLE()</code></td>
<td><code>__HAL_RCC_GPIOI_CLK_DISABLE()</code></td>
</tr>
<tr>
<td><code>#define SD_DETECT_EXTI_IRQHandler EXTI15_10_IRQHandler</code></td>
<td></td>
</tr>
<tr>
<td><code>#define TS_INT_PIN ((uint32_t)GPIO_PIN_13)</code></td>
<td>Touch screen interrupt signal.</td>
</tr>
<tr>
<td><code>#define TS_INT_GPIO_PORT ((GPIO_TypeDef*)GPIOI)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define TS_INT_GPIO_CLK_ENABLE()</code></td>
<td><code>__HAL_RCC_GPIOI_CLK_ENABLE()</code></td>
</tr>
<tr>
<td><code>#define TS_INT_GPIO_CLK_DISABLE()</code></td>
<td><code>__HAL_RCC_GPIOI_CLK_DISABLE()</code></td>
</tr>
<tr>
<td><code>#define TS_INT_EXTI_IRQHandler EXTI15_10_IRQHandler</code></td>
<td></td>
</tr>
<tr>
<td><code>#define TS_I2C_ADDRESS ((uint16_t)0x54)</code></td>
<td>Touch Screen FT6206 Slave I2C address 1.</td>
</tr>
<tr>
<td><code>#define TS_I2C_ADDRESS_A02 ((uint16_t)0x70)</code></td>
<td>Touch Screen FT6336G Slave I2C address 2.</td>
</tr>
<tr>
<td><code>#define AUDIO_I2C_ADDRESS ((uint16_t)0x34)</code></td>
<td>Audio I2C Slave address.</td>
</tr>
<tr>
<td><code>#define EEPROM_I2C_ADDRESS_A01 ((uint16_t)0xA0)</code></td>
<td>EEPROM I2C Slave address 1.</td>
</tr>
<tr>
<td><code>#define EEPROM_I2C_ADDRESS_A02 ((uint16_t)0xA6)</code></td>
<td>EEPROM I2C Slave address 2.</td>
</tr>
<tr>
<td><code>#define DISCOVERY_AUDIO_I2Cx</code> I2C4</td>
<td>User can use this section to tailor I2C4/I2C4 instance used and</td>
</tr>
<tr>
<td><code>#define DISCOVERY_AUDIO_I2Cx_CLK_ENABLE()</code></td>
<td><code>__HAL_RCC_I2C4_CLK_ENABLE()</code></td>
</tr>
<tr>
<td><code>#define DISCOVERY_AUDIO_I2Cx_SCL_GPIO_CLK_ENABLE()</code></td>
<td></td>
</tr>
<tr>
<td><code>#define DISCOVERY_AUDIO_I2Cx_SDA_GPIO_CLK_ENABLE()</code></td>
<td></td>
</tr>
<tr>
<td><code>#define DISCOVERY_AUDIO_I2Cx_SIOMUX_GPIO_CLK_ENABLE()</code></td>
<td></td>
</tr>
</tbody>
</table>
#define DISCOVERY_AUDIO_I2Cx_FORCE_RESET() __HAL_RCC_I2C4_FORCE_RESET()
#define DISCOVERY_AUDIO_I2Cx_RELEASE_RESET() __HAL_RCC_I2C4_RELEASE_RESET()

Definition for I2C4 Pins.

#define DISCOVERY_AUDIO_I2Cx_SCL_PIN GPIO_PIN_12
#define DISCOVERY_AUDIO_I2Cx_SCL_AF GPIO_AF4_I2C4
#define DISCOVERY_AUDIO_I2Cx_SCL_GPIO_PORT GPIOD
#define DISCOVERY_AUDIO_I2Cx_SDA_PIN GPIO_PIN_7
#define DISCOVERY_AUDIO_I2Cx_SDA_AF GPIO_AF11_I2C4
#define DISCOVERY_AUDIO_I2Cx_SDA_GPIO_PORT GPIOB

#define DISCOVERY_AUDIO_I2Cx_EV_IRQn I2C4_EV_IRQn
Definition of I2C4 interrupt requests.

#define DISCOVERY_AUDIO_I2Cx_ER_IRQn I2C4_ER_IRQn

#define DISCOVERY_EXT_I2Cx I2C1
User can use this section to tailor I2C1/I2C1 instance used and associated resources.

#define DISCOVERY_EXT_I2Cx_CLK_ENABLE() __HAL_RCC_I2C1_CLK_ENABLE()
#define DISCOVERY_DMAx_CLK_ENABLE() __HAL_RCC_DMA1_CLK_ENABLE()
#define DISCOVERY_EXT_I2Cx_SCL_SDA_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_ENABLE()
#define DISCOVERY_EXT_I2Cx_FORCE_RESET() __HAL_RCC_I2C1_FORCE_RESET()
#define DISCOVERY_EXT_I2Cx_RELEASE_RESET() __HAL_RCC_I2C1_RELEASE_RESET()

#define DISCOVERY_EXT_I2Cx_SCL_PIN GPIO_PIN_8
Definition for I2C1 Pins.

#define DISCOVERY_EXT_I2Cx_SCL_SDA_GPIO_PORT GPIOB
#define DISCOVERY_EXT_I2Cx_SCL_SDA_AF GPIO_AF4_I2C1
#define DISCOVERY_EXT_I2Cx_SDA_PIN GPIO_PIN_9

#define DISCOVERY_EXT_I2Cx_EV_IRQn I2C1_EV_IRQn
Definition of I2C interrupt requests.

#define DISCOVERY_EXT_I2Cx_ER_IRQn I2C1_ER_IRQn
Define Documentation

```c
#define AUDIO_I2C_ADDRESS  ((uint16_t)0x34)
```

Audio I2C Slave address.

Definition at line 221 of file `stm32f769i_discovery.h`.

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

```c
#define DISCOVERY_AUDIO_I2Cx  I2C4
```

User can use this section to tailor I2C4/I2C4 instance used and associated resources (audio codec).

Definition for I2C4 clock resources

Definition at line 238 of file `stm32f769i_discovery.h`.

Referenced by `I2Cx_Init()`.

```c
#define DISCOVERY_AUDIO_I2Cx_CLK_ENABLE ( ) __HAL_RCC_I2C4_CLK_ENABLE()
```

Definition at line 239 of file `stm32f769i_discovery.h`.

Referenced by `I2Cx_MspInit()`.

```c
#define DISCOVERY_AUDIO_I2Cx_ERIRQ  I2C4_ER_IRQn
```

Definition at line 257 of file stm32f769i_discovery.h.
Referenced by I2Cx_MspInit().

```c
#define DISCOVERY_AUDIO_I2Cx_EVIRQn  I2C4_EVIRQn
```
Definition of I2C4 interrupt requests.
Definition at line 256 of file stm32f769i_discovery.h.
Referenced by I2Cx_MspInit().

```c
#define DISCOVERY_AUDIO_I2Cx_FORCE_RESET()  __HAL_RCC_I2C4_FORCE_RESET()
```
Definition at line 243 of file stm32f769i_discovery.h.
Referenced by I2Cx_MspInit().

```c
#define DISCOVERY_AUDIO_I2Cx_RELEASE_RESET()  __HAL_RCC_I2C4_RELEASE_RESET()
```
Definition at line 244 of file stm32f769i_discovery.h.
Referenced by I2Cx_MspInit().

```c
#define DISCOVERY_AUDIO_I2Cx_SCL_AF        GPIO_AF4_I2C4
```
Definition at line 249 of file stm32f769i_discovery.h.
Referenced by I2Cx_MspInit().

```c
#define DISCOVERY_AUDIO_I2Cx_SCL_GPIO_CLK_ENABLE()  __HAL_RCC_GPIOD_CLK_ENABLE()
```
Definition at line 240 of file stm32f769i_discovery.h.
Referenced by `I2Cx_MspInit()`.

```c
#define DISCOVERY_AUDIO_I2Cx_SCL_GPIO_PORT GPIOD
```

Definition at line 250 of file `stm32f769i_discovery.h`.

Referenced by `I2Cx_MspInit()`.

```c
#define DISCOVERY_AUDIO_I2Cx_SCL_PIN GPIO_PIN_12
```

Definition for `I2C4` Pins.

PD12

Definition at line 248 of file `stm32f769i_discovery.h`.

Referenced by `I2Cx_MspInit()`.

```c
#define DISCOVERY_AUDIO_I2Cx_SDA_AF GPIO_AF11_I2C4
```

Definition at line 252 of file `stm32f769i_discovery.h`.

Referenced by `I2Cx_MspInit()`.

```c
#define DISCOVERY_AUDIO_I2Cx_SDA_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOB_CLK_ENABLE()
```

Definition at line 241 of file `stm32f769i_discovery.h`.

Referenced by `I2Cx_MspInit()`.

```c
#define DISCOVERY_AUDIO_I2Cx_SDA_GPIO_PORT GPIOB
```

Definition at line 253 of file `stm32f769i_discovery.h`. 
Referenced by \texttt{I2Cx\_MspInit()}. 

\begin{verbatim}
#define DISCOVERY\_AUDIO\_I2Cx\_SDA\_PIN GPIO\_PIN\_7 

PB7

Definition at line 251 of file \texttt{stm32f769i\_discovery.h}. 

Referenced by \texttt{I2Cx\_MspInit()}. 
\end{verbatim}

\begin{verbatim}
#define DISCOVERY\_DMAx\_CLK\_ENABLE () __HAL\_RCC\_DMA\_1\_ENABLE() 

Definition at line 266 of file \texttt{stm32f769i\_discovery.h}. 

Referenced by \texttt{I2Cx\_MspInit()}. 
\end{verbatim}

\begin{verbatim}
#define DISCOVERY\_EXT\_I2Cx \texttt{I2C1} 

User can use this section to tailor \texttt{I2C1/I2C1} instance used and associated resources. 

Definition for \texttt{I2C1} clock resources 

Definition at line 264 of file \texttt{stm32f769i\_discovery.h}. 

Referenced by \texttt{I2Cx\_Init()}. 
\end{verbatim}

\begin{verbatim}
#define DISCOVERY\_EXT\_I2Cx\_CLK\_ENABLE () __HAL\_RCC\_I2\_1\_ENABLE() 

Definition at line 265 of file \texttt{stm32f769i\_discovery.h}. 

Referenced by \texttt{I2Cx\_MspInit()}. 
\end{verbatim}

\begin{verbatim}
#define DISCOVERY\_EXT\_I2Cx\_ER\_IRQn \texttt{I2C1\_ER\_IRQn} 
\end{verbatim}
Definition at line 282 of file `stm32f769i_discovery.h`.
Referenced by `I2Cx_MspInit()`.

```c
#define DISCOVERY_EXT_I2Cx_EV_IRQn I2C1_EV_IRQn
```
Definition of I2C interrupt requests.
Definition at line 281 of file `stm32f769i_discovery.h`.
Referenced by `I2Cx_MspInit()`.

```c
#define DISCOVERY_EXT_I2Cx_FORCE_RESET () __HAL_RCC_I2C1_FORCE_RESET()
```
Definition at line 269 of file `stm32f769i_discovery.h`.
Referenced by `I2Cx_MspInit()`.

```c
#define DISCOVERY_EXT_I2Cx_RELEASE_RESET () __HAL_RCC_I2C1_RELEASE_RESET()
```
Definition at line 270 of file `stm32f769i_discovery.h`.
Referenced by `I2Cx_MspInit()`.

```c
#define DISCOVERY_EXT_I2Cx_SCL_PIN GPIO_PIN_8
```
Definition for I2C1 Pins.
PB8
Definition at line 274 of file `stm32f769i_discovery.h`.
Referenced by `I2Cx_MspInit()`.
```c
#define DISCOVERY_EXT_I2Cx_SCL_SDA_AF   GPIO_AF4_I2C1

Definition at line 276 of file stm32f769i_discovery.h.
Referenced by I2Cx_MspInit().

#define DISCOVERY_EXT_I2Cx_SCL_SDA_GPIO_CLK_ENABLE ( )

Definition at line 267 of file stm32f769i_discovery.h.
Referenced by I2Cx_MspInit().

#define DISCOVERY_EXT_I2Cx_SCL_SDA_GPIO_PORT   GPIOB

Definition at line 275 of file stm32f769i_discovery.h.
Referenced by I2Cx_MspInit().

#define DISCOVERY_EXT_I2Cx_SDA_PIN   GPIO_PIN_9

PB9
Definition at line 277 of file stm32f769i_discovery.h.
Referenced by I2Cx_MspInit().

#define EEPROM_I2C_ADDRESS_A01   ((uint16_t)0xA0)

EEPROM I2C Slave address 1.
Definition at line 226 of file stm32f769i_discovery.h.
Referenced by `BSP_EEPROM_Init()`.

```c
#define EEPROM_I2C_ADDRESS_A02 ((uint16_t)0xA6)
```

EEPROM I2C Slave address 2.

Definition at line 231 of file `stm32f769i_discovery.h`.

Referenced by `BSP_EEPROM_Init()`.

```c
#define OTG_HS_OVER_CURRENT_PIN GPIO_PIN_4
```

USB OTG HS Over Current signal.

Definition at line 186 of file `stm32f769i_discovery.h`.

```c
#define OTG_HS_OVER_CURRENT_PORT GPIOD
```

Definition at line 187 of file `stm32f769i_discovery.h`.

```c
#define OTG_HS_OVER_CURRENT_PORT_CLK_ENABLE () __HAL_RCC_GPIOD_CLK_ENABLE()
```

Definition at line 188 of file `stm32f769i_discovery.h`.

```c
#define SD_DETECT_EXTI_IRQn EXTI15_10_IRQn
```

Definition at line 197 of file `stm32f769i_discovery.h`.

Referenced by `BSP_SD_ITConfig()`.

```c
#define SD_DETECT_GPIO_CLK_DISABLE () __HAL_RCC_GPIOI_CLK_DISABLE()
```

Definition at line 197 of file `stm32f769i_discovery.h`.

Referenced by `BSP_SD_ITConfig()`.
#define SD_DETECT_GPIO_CLK_ENABLE () __HAL_RCC_GPIOI_CLK_ENABLE()

Definition at line 196 of file stm32f769i_discovery.h.

Referenced by BSP_SD_Detect_MspInit().

#define SD_DETECT_GPIO_PORT ((GPIO_TypeDef*)GPIOI)

Definition at line 195 of file stm32f769i_discovery.h.

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

#define SD_DETECT_PIN ((uint32_t)GPIO_PIN_15)

SD-detect signal.

Definition at line 194 of file stm32f769i_discovery.h.

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

#define TS_I2C_ADDRESS ((uint16_t)0x54)

TouchScreen FT6206 Slave I2C address 1.

Definition at line 211 of file stm32f769i_discovery.h.

Referenced by BSP_TS_Init().

#define TS_I2C_ADDRESS_A02 ((uint16_t)0x70)
TouchScreen FT6336G Slave I2C address 2.

Definition at line 216 of file `stm32f769i_discovery.h`.

Referenced by `BSP_TS_Init()`.

```c
#define TS_INT_EXTI_IRQn EXTI15_10_IRQn
```

Definition at line 206 of file `stm32f769i_discovery.h`.

Referenced by `BSP_TS_ITConfig()`.

```c
#define TS_INT_GPIO_CLK_DISABLE (__HAL_RCC_GPIOI_CLK_DISABLE())
```

Definition at line 205 of file `stm32f769i_discovery.h`.

Referenced by `BSP_TS_INT_MspInit()`.

```c
#define TS_INT_GPIO_CLK_ENABLE (__HAL_RCC_GPIOI_CLK_ENABLE())
```

Definition at line 204 of file `stm32f769i_discovery.h`.

Referenced by `BSP_TS_INT_MspInit()`, and `BSP_TS_ITConfig()`.

```c
#define TS_INT_GPIO_PORT ((GPIO_TypeDef*)GPIOI)
```

Definition at line 203 of file `stm32f769i_discovery.h`.

Referenced by `BSP_TS_INT_MspInit()`, and `BSP_TS_ITConfig()`.

```c
#define TS_INT_PIN ((uint32_t)GPIO_PIN_13)
```

Touch screen interrupt signal.
Definition at line 202 of file stm32f769i_discovery.h.

Referenced by BSP_TS_INT_MspInit(), and BSP_TS_ITConfig().
# STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **LOW_LEVEL** Private Function Prototypes

  ```c
  STM32F769I_DISCOVERY_LOW_LEVEL
  ```
### Functions

<table>
<thead>
<tr>
<th>void</th>
<th>AUDIO_IO_Init (void)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initializes Audio low level.</td>
</tr>
<tr>
<td>void</td>
<td>AUDIO_IO_DeInit (void)</td>
</tr>
<tr>
<td></td>
<td>DeInitializes Audio low level.</td>
</tr>
<tr>
<td>void</td>
<td>EEPROM_IO_Init (void)</td>
</tr>
<tr>
<td></td>
<td>Initializes peripherals used by the I2C EEPROM driver.</td>
</tr>
</tbody>
</table>
### Function Documentation

**void AUDIO_IO_DeInit ( void )**

DeInitializes Audio low level.

Definition at line 563 of file `stm32f769i_discovery.c`.

**void AUDIO_IO_Init ( void )**

Initializes Audio low level.

Definition at line 555 of file `stm32f769i_discovery.c`.

References `hI2cAudioHandler`, and `I2Cx_Init()`.

**void EEPROM_IO_Init ( void )**

Initializes peripherals used by the I2C EEPROM driver.

Definition at line 621 of file `stm32f769i_discovery.c`.

References `hI2cExtHandler`, and `I2Cx_Init()`.

Referenced by `BSP_EEPROM_Init()`.
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**BSP Audio Sample Rate**

STM32F769I_DISCOVERY_AUDIO Exported Constants
#define BSP_AUDIO_FREQUENCY_96K SAI_AUDIO_FREQUENCY_96K
#define BSP_AUDIO_FREQUENCY_48K SAI_AUDIO_FREQUENCY_48K
#define BSP_AUDIO_FREQUENCY_44K SAI_AUDIO_FREQUENCY_44K
#define BSP_AUDIO_FREQUENCY_32K SAI_AUDIO_FREQUENCY_32K
#define BSP_AUDIO_FREQUENCY_22K SAI_AUDIO_FREQUENCY_22K
#define BSP_AUDIO_FREQUENCY_16K SAI_AUDIO_FREQUENCY_16K
#define BSP_AUDIO_FREQUENCY_11K SAI_AUDIO_FREQUENCY_11K
#define BSP_AUDIO_FREQUENCY_8K SAI_AUDIO_FREQUENCY_8K
Define Documentation

```c
#define BSP_AUDIO_FREQUENCY_11K   SAI_AUDIO_FREQUENCY
```
Definition at line 85 of file `stm32f769i_discovery_audio.h`.

```c
#define BSP_AUDIO_FREQUENCY_16K   SAI_AUDIO_FREQUENCY
```
Definition at line 84 of file `stm32f769i_discovery_audio.h`.

```c
#define BSP_AUDIO_FREQUENCY_22K   SAI_AUDIO_FREQUENCY
```
Definition at line 83 of file `stm32f769i_discovery_audio.h`.

```c
#define BSP_AUDIO_FREQUENCY_32K   SAI_AUDIO_FREQUENCY
```
Definition at line 82 of file `stm32f769i_discovery_audio.h`.

```c
#define BSP_AUDIO_FREQUENCY_44K   SAI_AUDIO_FREQUENCY
```
Definition at line 81 of file `stm32f769i_discovery_audio.h`.

```c
#define BSP_AUDIO_FREQUENCY_48K   SAI_AUDIO_FREQUENCY
```
Definition at line 80 of file `stm32f769i_discovery_audio.h`.

```c
#define BSP_AUDIO_FREQUENCY_8K    SAI_AUDIO_FREQUENCY
```
Definition at line 86 of file `stm32f769i_discovery_audio.h`.
#define BSP_AUDIO_FREQUENCY_96K SAI_AUDIO_FREQUENCY

Definition at line 79 of file stm32f769i_discovery_audio.h.
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**STM32F769I_DISCOVERY_AUDIO_In Private Functions**

STM32F769I_DISCOVERY AUDIO
### Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td><strong>BSP_AUDIO_IN_Init</strong> (uint32_t AudioFreq, uint32_t BitRes, uint32_t ChnlNbr)</td>
<td>Initialize wave recording.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_AUDIO_IN_InitEx</strong> (uint16_t InputDevice, uint32_t AudioFreq, uint32_t BitRes, uint32_t ChnlNbr)</td>
<td>Initialize wave recording.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_AUDIO_IN_AllocScratch</strong> (int32_t *pScratch, uint32_t size)</td>
<td>Allocate channel buffer scratch.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_AUDIO_IN_GetChannelNumber</strong> (void)</td>
<td>Return audio in channel number.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_AUDIO_IN_Record</strong> (uint16_t *pbuf, uint32_t size)</td>
<td>Start audio recording.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_AUDIO_IN_Stop</strong> (void)</td>
<td>Stop audio recording.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_AUDIO_IN_Pause</strong> (void)</td>
<td>Pause the audio file stream.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_AUDIO_IN_Resume</strong> (void)</td>
<td>Resume the audio file stream.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_AUDIO_IN_DeInit</strong> (void)</td>
<td>Deinit the audio IN peripherals.</td>
</tr>
<tr>
<td>void</td>
<td><strong>HAL_DFSDM_FilterRegConvCpltCallback</strong> (DFSDM_Filter_HandleTypeDef *hdfsdm_filter)</td>
<td>Regular conversion complete callback.</td>
</tr>
<tr>
<td>void</td>
<td><strong>HAL_DFSDM_FilterRegConvHalfCpltCallback</strong> (DFSDM_Filter_HandleTypeDef *hdfsdm_filter)</td>
<td>Half regular conversion complete callback.</td>
</tr>
<tr>
<td>void</td>
<td><strong>HAL_SAI_RxHalfCpltCallback</strong> (SAI_HandleTypeDef *hsai)</td>
<td>Half reception complete callback.</td>
</tr>
<tr>
<td>void</td>
<td><strong>HAL_SAI_RxCpltCallback</strong> (SAI_HandleTypeDef *hsai)</td>
<td></td>
</tr>
</tbody>
</table>
Reception complete callback.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>__weak void</strong> BSP_AUDIO_IN_TransferComplete_CallBack (void)</td>
<td>User callback when record buffer is filled.</td>
</tr>
<tr>
<td><strong>__weak void</strong> BSP_AUDIO_IN_HalfTransfer_CallBack (void)</td>
<td>Manages the DMA Half Transfer complete event.</td>
</tr>
<tr>
<td><strong>__weak void</strong> BSP_AUDIO_IN_Error_CallBack (void)</td>
<td>Audio IN Error callback function.</td>
</tr>
<tr>
<td><strong>__weak void</strong> BSP_AUDIO_IN_MspInit (void)</td>
<td>Initialize BSP_AUDIO_IN MSP.</td>
</tr>
<tr>
<td><strong>__weak void</strong> BSP_AUDIO_IN_MspDeInit (void)</td>
<td>DeInitialize BSP_AUDIO_IN MSP.</td>
</tr>
<tr>
<td><strong>__weak void</strong> BSP_AUDIO_IN_ClockConfig (DFSDM_Filter_HandleTypeDef *hdfsdm_filter, uint32_t AudioFreq, void *Params)</td>
<td>Clock Config.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>static uint8_t</strong> DFSDMx_Init (uint32_t AudioFreq)</td>
<td>Initialize the Digital Filter for Sigma-Delta Modulators interface (DFSDM).</td>
</tr>
<tr>
<td><strong>static uint8_t</strong> DFSDMx_DeInit (void)</td>
<td>De-initialize the Digital Filter for Sigma-Delta Modulators interface (DFSDM).</td>
</tr>
<tr>
<td><strong>static void</strong> DFSDMx_ChannelMspInit (void)</td>
<td>Initialize the DFSDM channel MSP.</td>
</tr>
<tr>
<td><strong>static void</strong> DFSDMx_ChannelMspDeInit (void)</td>
<td>DeInitialize the DFSDM channel MSP.</td>
</tr>
<tr>
<td><strong>static void</strong> DFSDMx_FilterMspInit (void)</td>
<td>Initialize the DFSDM filter MSP.</td>
</tr>
<tr>
<td><strong>static void</strong> DFSDMx_FilterMspDeInit (void)</td>
<td>DeInitialize the DFSDM filter MSP.</td>
</tr>
</tbody>
</table>
Function Documentation

```c
uint8_t BSP_AUDIO_IN_AllocScratch ( int32_t * pScratch,
                                  uint32_t size
)
```

Allocate channel buffer scratch.

**Parameters:**
- **pScratch**: pointer to scratch tables.
- **size**: size of scratch buffer

Definition at line 1194 of file stm32f769i_discovery_audio.c.

References **AUDIO_OK**, **AudioIn_ChannelNumber**, **pScratchBuff**, and **ScratchSize**.

```c
__weak void BSP_AUDIO_IN_ClockConfig ( DFSDM_Filter_HandleTypeDef * hdfsdm_filter,
                                       uint32_t AudioFreq,
                                       void * Params
)
```

Clock Config.

**Parameters:**
- **hdfsdm_filter**: might be required to set audio peripheral predivider if any.
- **AudioFreq**: Audio frequency used to play the audio stream.
- **Params**: 

**Note:**
This API is called by **BSP_AUDIO_IN_Init**. Being __weak it can be overwritten by the application.
Return values:
  None

Definition at line 1680 of file stm32f769i_discovery_audio.c.

Referenced by BSP_AUDIO_IN_InitEx().


void BSP_AUDIO_IN_DeInit ( void )

Deinit the audio IN peripherals.

  Return values:
    None

Definition at line 1410 of file stm32f769i_discovery_audio.c.

References AudioIn_Device, BSP_AUDIO_IN_MspDeInit(), DFSDMx_DeInit(), INPUT_DEVICE_DIGITAL_MIC, and SAIx_In_DeInit().

__weak void BSP_AUDIO_IN_Error_CallBack ( void )

Audio IN Error callback function.

  Return values:
    None

Definition at line 1627 of file stm32f769i_discovery_audio.c.

Referenced by HAL_SAI_ErrorCallback().


uint8_t BSP_AUDIO_IN_GetChannelNumber ( void )

Return audio in channel number.
Return values:
  Number of channel
Definition at line 1213 of file stm32f769i_discovery_audio.c.
References AudioIn_ChannelNumber.

 weak void BSP_AUDIO_IN_HalfTransfer_CallBack ( void )

Manages the DMA Half Transfer complete event.

Return values:
  None
Definition at line 1616 of file stm32f769i_discovery_audio.c.
Referenced by HAL_DFSDM_FilterRegConvCpltCallback(), HAL_DFSDM_FilterRegConvHalfCpltCallback(), and HAL_SAI_RxHalfCpltCallback().

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

Initialize wave recording.

Parameters:
  AudioFreq,: Audio frequency to be configured for the DFSDM peripheral.
  BitRes,: Audio frequency to be configured for the DFSDM peripheral.
  ChnINbr,: Audio frequency to be configured for the DFSDM peripheral.
Return values:

**AUDIO_OK** if correct communication, else wrong communication

Definition at line 1117 of file `stm32f769i_discovery_audio.c`.

References `BSP_AUDIO_IN_InitEx()`, and `INPUT_DEVICE_DIGITAL_MIC`.

```c
uint8_t BSP_AUDIO_IN_InitEx ( uint16_t InputDevice,
                              uint32_t AudioFreq,
                              uint32_t BitRes,
                              uint32_t ChnlNbr )
```

Initialize wave recording.

**Parameters:**

- **InputDevice:** INPUT_DEVICE_DIGITAL_MIC or INPUT_DEVICE_ANALOG_MIC.
- **AudioFreq:** Audio frequency to be configured.
- **BitRes:** Audio bit resolution to be configured.
- **ChnlNbr:** Number of channel to be configured.

Return values:

**AUDIO_OK** if correct communication, else wrong communication

Definition at line 1130 of file `stm32f769i_discovery_audio.c`.

References `audio_drv`, **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_IN_SAIX**, **AUDIO_OK**, Audioln_ChannelNumber, AudioIn_Device, **BSP_AUDIO_IN_ClockConfig()**, **BSP_AUDIO_IN_MspInit()**, **BSP_AUDIO_OUT_ClockConfig()**, **BSP_AUDIO_OUT_MspInit()**, **DFSDMx_Init()**, **haudio_in_sai**,
Referenced by **BSP_AUDIO_IN_Init()**.

```c
__weak void BSP_AUDIO_IN_MspDeInit ( void )
```

Deinitialize BSP_AUDIO_IN MSP.

**Return values:**

None

Definition at line 1656 of file `stm32f769i_discovery_audio.c`.

References **AudioIn_Device**, **DFSDMx_ChannelMspDeInit()**, **DFSDMx_FilterMspDeInit()**, **haudio_in_sai**, **INPUT_DEVICE_DIGITAL_MIC**, and **SAI_AUDIO_IN_MspDeInit()**.

Referenced by **BSP_AUDIO_IN_DeInit()**.

```c
__weak void BSP_AUDIO_IN_MspInit ( void )
```

Initialize BSP_AUDIO_IN MSP.

**Return values:**

None

Definition at line 1637 of file `stm32f769i_discovery_audio.c`.

References **AudioIn_Device**, **DFSDMx_ChannelMspInit()**, **DFSDMx_FilterMspInit()**, **haudio_in_sai**, **INPUT_DEVICE_DIGITAL_MIC**, and **SAI_AUDIO_IN_MspInit()**.

Referenced by **BSP_AUDIO_IN_InitEx()**.
uint8_t BSP_AUDIO_IN_Pause (void)

Pause the audio file stream.

Return values:

   AUDIO_OK  if correct communication, else wrong communication

Definition at line 1338 of file stm32f769i_discovery_audio.c.

References AUDIO_ERROR, AUDIO_OK, AudioIn_ChannelNumber, hAudioInBottomLeftFilter, hAudioInBottomRightFilter, hAudioInTopLeftFilter, and hAudioInTopRightFilter.

uint8_t BSP_AUDIO_IN_Record (uint16_t * pbuf, uint32_t size)

Start audio recording.

Parameters:

   pbuf,: Main buffer pointer for the recorded data storing
   size,: Current size of the recorded buffer

Return values:

   AUDIO_OK  if correct communication, else wrong communication

Definition at line 1224 of file stm32f769i_discovery_audio.c.

References AppBuffHalf, AppBuffTrigger, AUDIO_ERROR, AUDIO_OK, AudioIn_ChannelNumber, AudioIn_Device, haudio_in_sai, hAudioInBottomLeftFilter, hAudioInBottomRightFilter, hAudioInTopLeftFilter, hAudioInTopRightFilter, INPUT_DEVICE_DIGITAL_MIC,
uint8_t BSP_AUDIO_IN_Resume ( void )

Resume the audio file stream.

**Return values:**

AUDIO_OK if correct communication, else wrong communication

Definition at line 1374 of file stm32f769i_discovery_audio.c.

References AUDIO_ERROR, AUDIO_OK, AudioIn_ChannelNumber, hAudioInButtomLeftFilter, hAudioInButtomRightFilter, hAudioInTopLeftFilter, hAudioInTopRightFilter, pScratchBuff, and ScratchSize.

uint8_t BSP_AUDIO_IN_Stop ( void )

Stop audio recording.

**Return values:**

AUDIO_OK if correct communication, else wrong communication

Definition at line 1277 of file stm32f769i_discovery_audio.c.

References AppBuffHalf, AppBuffTrigger, audio_drv, AUDIO_ERROR, AUDIO_I2C_ADDRESS, AUDIO_OK, AudioIn_ChannelNumber, AudioIn_Device, haudio_in_sai, hAudioInButtomLeftFilter, hAudioInButtomRightFilter, hAudioInTopLeftFilter, hAudioInTopRightFilter, and INPUT_DEVICE_DIGITAL_MIC.
__weak void BSP_AUDIO_IN_TransferComplete_CallBack ( void )

User callback when record buffer is filled.

Return values:
   None

Definition at line 1605 of file stm32f769i_discovery_audio.c.

Referenced by HAL_DFSDM_FilterRegConvCpltCallback(), HAL_DFSDM_FilterRegConvHalfCpltCallback(), and HAL_SAI_RxCpltCallback().

static void DFSDMx_ChannelMspDeInit ( void ) [static]

DeInitialize the DFSDM channel MSP.

Return values:
   None

Definition at line 2036 of file stm32f769i_discovery_audio.c.

References AUDIO_DFSDMx_CKOUT_DMIC_GPIO_PORT, AUDIO_DFSDMx_CKOUT_PIN, AUDIO_DFSDMx_DMIC_DATIN1_PIN, AUDIO_DFSDMx_DMIC_DATIN5_PIN, AUDIO_DFSDMx_DMIC_DATIN_GPIO_PORT, and AudioIn_ChannelNumber.

Referenced by BSP_AUDIO_IN_MspDeInit().

static void DFSDMx_ChannelMspInit ( void ) [static]

Initialize the DFSDM channel MSP.
Return values:

None

Definition at line 1999 of file stm32f769i_discovery_audio.c.

References AUDIO_DFSDMx_CKOUT_AF, AUDIO_DFSDMx_CKOUT_DMIC_GPIO_CLK_ENABLE, AUDIO_DFSDMx_CKOUT_DMIC_GPIO_PORT, AUDIO_DFSDMx_CKOUT_PIN, AUDIO_DFSDMx_CLK_ENABLE, AUDIO_DFSDMx_DMIC_DATIN1_PIN, AUDIO_DFSDMx_DMIC_DATIN5_PIN, AUDIO_DFSDMx_DMIC_DATIN_AF, AUDIO_DFSDMx_DMIC_DATIN_GPIO_CLK_ENABLE, AUDIO_DFSDMx_DMIC_DATIN_GPIO_PORT, and AudioIn_ChannelNumber.

Referenced by BSP_AUDIO_IN_MspInit().

static uint8_t DFSDMx_DeInit (void) [static]

De-initialize the Digital Filter for Sigma-Delta Modulators interface (DFSDM).

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line 1943 of file stm32f769i_discovery_audio.c.

References AUDIO_ERROR, AUDIO_OK, AudioIn_ChannelNumber, hAudioInButtomLeftChannel, hAudioInButtomLeftFilter, hAudioInButtomRightChannel, hAudioInButtomRightFilter, hAudioInTopLeftChannel, hAudioInTopLeftFilter, hAudioInTopRightChannel, and hAudioInTopRightFilter.

Referenced by BSP_AUDIO_IN_DeInit().
static void DFSDMx_FilterMspDeInit ( void ) [static]

Deinitialize the DFSDM filter MSP.

Return values:

None

Definition at line 2174 of file stm32f769i_discovery_audio.c.

References AudioIn_ChannelNumber, hDmaButtomLeft, hDmaButtomRight, hDmaTopLeft, and hDmaTopRight.

Referenced by BSP_AUDIO_IN_MspDeInit().

---

static void DFSDMx_FilterMspInit ( void ) [static]

Initialize the DFSDM filter MSP.

Return values:

None

Definition at line 2060 of file stm32f769i_discovery_audio.c.

References AUDIO_DFSDMx_CLK_ENABLE, AUDIO_DFSDMx_DMAx_BUTTOM_LEFT_IRQ, AUDIO_DFSDMx_DMAx_BUTTOM_LEFT_STREAM, AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_IRQ, AUDIO_DFSDMx_DMAx_BUTTOM_RIGHT_STREAM, AUDIO_DFSDMx_DMAx_CHANNEL, AUDIO_DFSDMx_DMAx_CLK_ENABLE, AUDIO_DFSDMx_DMAx_MEM_DATA_SIZE, AUDIO_DFSDMx_DMAx_PERIPH_DATA_SIZE, AUDIO_DFSDMx_DMAx_TOP_LEFT_IRQ, AUDIO_DFSDMx_DMAx_TOP_LEFT_STREAM, AUDIO_DFSDMx_DMAx_TOP_RIGHT_IRQ,
Referenced by `BSP_AUDIO_IN_MspInit()`.

```c
static uint8_t DFSDMx_Init ( uint32_t AudioFreq )
```

Initialize the Digital Filter for Sigma-Delta Modulators interface (DFSDM).

**Parameters:**

- **AudioFreq:** Audio frequency to be used to set correctly the DFSDM peripheral.

**Note:**

Channel output Clock Divider and Filter Oversampling are calculated as follow:

- Clock Divider = \( \frac{\text{CLK} (\text{input DFSDM})}{\text{CLK} (\text{micro})} \) with 1MHZ < CLK (micro) < 3.2MHZ (TYP 2.4MHZ for MP34DT01TR)
- Oversampling = \( \frac{\text{CLK} (\text{input DFSDM})}{(\text{Clock Divider} \times \text{AudioFreq})} \)

**Return values:**

- **AUDIO_OK** if correct communication, else wrong communication

Definition at line 1733 of file `stm32f769i_discovery_audio.c`.

References `AUDIO_DFSDMx_BUTTOM_LEFT_CHANNEL`, `AUDIO_DFSDMx_BUTTOM_LEFT_FILTER`, `AUDIO_DFSDMx_BUTTOM_RIGHT_CHANNEL`, `AUDIO_DFSDMx_BUTTOM_RIGHT_FILTER`, `AUDIO_DFSDMx_TOP_LEFT_CHANNEL`,

```c
```
void HAL_DFSDM_FilterRegConvCpltCallback (DFSDM_Filter_Han

Regular conversion complete callback.

Note:
In interrupt mode, user has to read conversion value in this
function using HAL_DFSDM_FilterGetRegularValue.

Parameters:
  hdfsdm_filter : DFSDM filter handle.

Return values:
  None

Definition at line 1431 of file stm32f769i_discovery_audio.c.

References AppBuffHalf, AppBuffTrigger, 
AudioIn_ChannelNumber,
BSP_AUDIO_IN_HalfTransfer_CallBack(), 
BSP_AUDIO_IN_TransferComplete_CallBack(), 
DmaButtomLeftRecCplt, DmaButtomRightRecCplt, 
DmaTopLeftRecCplt, DmaTopRightRecCplt, 
hAudioInButtomLeftFilter, hAudioInTopLeftFilter,
void HAL_DFSDM_FilterRegConvHalfCpltCallback ( DFSDM_Filter handle)

Half regular conversion complete callback.

Parameters:
  handle : DFSDM filter handle.

Return values:
  None

Definition at line 1507 of file stm32f769i_discovery_auido.c.

References AppBuffHalf, AppBuffTrigger, AudioIn_ChannelNumber, BSP_AUDIO_IN_HalfTransfer_CallBack(), BSP_AUDIO_IN_TransferComplete_CallBack(), DmaButtomLeftRecHalfCplt, DmaButtomRightRecHalfCplt, DmaTopLeftRecHalfCplt, DmaTopRightRecHalfCplt, hAudioInButtomLeftFilter, hAudioInTopLeftFilter, hAudioInTopRightFilter, AUDIOIN_TypeDef::pRecBuf, pScratchBuff, AUDIOIN_TypeDef::RecSize, SaturaLH, and ScratchSize.

void HAL_SAI_RxCpltCallback ( SAI_HandleTypeDef * hsai )

Reception complete callback.

Parameters:
  hsai : SAI handle.

Return values:
void HAL_SAI_RxHalfCpltCallback ( SAI_HandleTypeDef * hsai )

Half reception complete callback.

**Parameters:**

- **hsai**: SAI handle.

**Return values:**

**None**

Definition at line 1583 of file stm32f769i_discovery_audio.c.

References **BSP_AUDIO_IN_HalfTransfer_CallBack()**.
# STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## STM32F769I_DISCOVERY_AUDIO_IN Exported Functions

STM32F769I_DISCOVERY AUDIO
**Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>uint8_t</strong> BSP_AUDIO_IN_Init (uint32_t AudioFreq, uint32_t BitRes, uint32_t ChnlNbr)</td>
<td>Initialize wave recording.</td>
</tr>
<tr>
<td><strong>uint8_t</strong> BSP_AUDIO_IN_InitEx (uint16_t InputDevice, uint32_t AudioFreq, uint32_t BitRes, uint32_t ChnlNbr)</td>
<td>Initialize wave recording.</td>
</tr>
<tr>
<td><strong>uint8_t</strong> BSP_AUDIO_IN_AllocScratch (int32_t *pScratch, uint32_t size)</td>
<td>Allocate channel buffer scratch.</td>
</tr>
<tr>
<td><strong>uint8_t</strong> BSP_AUDIO_IN_GetChannelNumber (void)</td>
<td>Return audio in channel number.</td>
</tr>
<tr>
<td><strong>void</strong> BSP_AUDIO_IN_DeInit (void)</td>
<td>Deinit the audio IN peripherals.</td>
</tr>
<tr>
<td><strong>uint8_t</strong> BSP_AUDIO_IN_Record (uint16_t *pData, uint32_t Size)</td>
<td>Start audio recording.</td>
</tr>
<tr>
<td><strong>uint8_t</strong> BSP_AUDIO_IN_Stop (void)</td>
<td>Stop audio recording.</td>
</tr>
<tr>
<td><strong>uint8_t</strong> BSP_AUDIO_IN_Pause (void)</td>
<td>Pause the audio file stream.</td>
</tr>
<tr>
<td><strong>uint8_t</strong> BSP_AUDIO_IN_Resume (void)</td>
<td>Resume the audio file stream.</td>
</tr>
<tr>
<td><strong>void</strong> BSP_AUDIO_IN_TransferComplete_CallBack (void)</td>
<td>User callback when record buffer is filled.</td>
</tr>
<tr>
<td><strong>void</strong> BSP_AUDIO_IN_HalfTransfer_CallBack (void)</td>
<td>Manages the DMA Half Transfer complete event.</td>
</tr>
<tr>
<td><strong>void</strong> BSP_AUDIO_IN_Error_CallBack (void)</td>
<td>Audio IN Error callback function.</td>
</tr>
<tr>
<td><strong>void</strong> BSP_AUDIO_IN_ClockConfig (DFSDM_Filter_HandleTypeDef *hdfsdm_filter, uint32_t AudioFreq, void *Params)</td>
<td>Clock Config.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_AUDIO_IN_MspInit</strong> (void)</td>
</tr>
<tr>
<td>-------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td></td>
<td>Initialize BSP_AUDIO_IN MSP.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_AUDIO_IN_MspDeInit</strong> (void)</td>
</tr>
<tr>
<td></td>
<td>DeInitialize BSP_AUDIO_IN MSP.</td>
</tr>
</tbody>
</table>
Function Documentation

```c
uint8_t BSP_AUDIO_IN_AllocScratch ( int32_t * pScratch,
                                  uint32_t size )
```

Allocate channel buffer scratch.

**Parameters:**
- `pScratch` : pointer to scratch tables.
- `size` : of scratch buffer

Definition at line `1194` of file `stm32f769i_discovery_audio.c`.

References `AUDIO_OK`, `AudioIn_ChannelNumber`, `pScratchBuff`, and `ScratchSize`.

```c
void BSP_AUDIO_IN_ClockConfig ( DFSDM_Filter_HandleTypeDef hdfsdm_filter,
                                uint32_t AudioFreq,
                                void * Params )
```

Clock Config.

**Parameters:**
- `hdfsdm_filter` : might be required to set audio peripheral predivider if any.
- `AudioFreq` : Audio frequency used to play the audio stream.
- `Params` : 

**Note:**
This API is called by `BSP_AUDIO_IN_Init()` Being `__weak` it can be overwritten by the application.
**Return values:**

*None*

Definition at line 1680 of file `stm32f769i_discovery_audio.c`.

Referenced by `BSP_AUDIO_IN_InitEx()`.

```c
void BSP_AUDIO_IN_DeInit ( void )
```

Deinit the audio IN peripherals.

**Return values:**

*None*

Definition at line 1410 of file `stm32f769i_discovery_audio.c`.

References `AudioIn_Device`, `BSP_AUDIO_IN_MspDeInit()`, `DFSDMx_DeInit()`, `INPUT_DEVICE_DIGITAL_MIC`, and `SAIx_In_DeInit()`.

```c
void BSP_AUDIO_IN_Error_CallBack ( void )
```

Audio IN Error callback function.

**Return values:**

*None*

Definition at line 1627 of file `stm32f769i_discovery_audio.c`.

Referenced by `HAL_SAI_ErrorCallback()`.

```c
uint8_t BSP_AUDIO_IN_GetChannelNumber ( void )
```

Return audio in channel number.
Return values:

**Number** of channel

Definition at line **1213** of file `stm32f769i_discovery_audio.c`.

References **AudioIn_ChannelNumber**.

```c
void BSP_AUDIO_IN_HalfTransfer_CallBack ( void )
```

Manages the DMA Half Transfer complete event.

Return values:

**None**

Definition at line **1616** of file `stm32f769i_discovery_audio.c`.

Referenced by **HAL_DFSDM_FilterRegConvCpltCallback()**, **HAL_DFSDM_FilterRegConvHalfCpltCallback()**, and **HAL_SAI_RxHalfCpltCallback()**.

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

Initialize wave recording.

**Parameters:**

- **AudioFreq**: Audio frequency to be configured for the DFSDM peripheral.
- **BitRes**: Audio frequency to be configured for the DFSDM peripheral.
- **ChnlNbr**: Audio frequency to be configured for the DFSDM peripheral.
Return values:

   **AUDIO_OK** if correct communication, else wrong communication

Definition at line **1117** of file **stm32f769i_discovery_audio.c**.

References **BSP_AUDIO_IN_InitEx()**, and **INPUT_DEVICE_DIGITAL_MIC**.

```c
uint8_t BSP_AUDIO_IN_InitEx ( uint16_t InputDevice,
                             uint32_t AudioFreq,
                             uint32_t BitRes,
                             uint32_t ChnlNbr )
```

Initialize wave recording.

**Parameters:**

- **InputDevice:**  INPUT_DEVICE_DIGITAL_MIC or INPUT_DEVICE_ANALOG_MIC.
- **AudioFreq:**  Audio frequency to be configured.
- **BitRes:**  Audio bit resolution to be configured.
- **ChnlNbr:**  Number of channel to be configured.

Return values:

   **AUDIO_OK** if correct communication, else wrong communication

Definition at line **1130** of file **stm32f769i_discovery_audio.c**.

References **audio_drv**, **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_IN_SAIx**, **AUDIO_OK**, **Audioln_ChannelNumber**, **AudioIn_Device**, **BSP_AUDIO_IN_ClockConfig()**, **BSP_AUDIO_IN_MspInit()**, **BSP_AUDIO_OUT_ClockConfig()**, **BSP_AUDIO_OUT_MspInit()**, **DFSDMx_Init()**, **haudio_in_sai**,
Referenced by **BSP_AUDIO_IN_Init()**.

```c
void BSP_AUDIO_IN_MspDeInit ( void )
```

DeInitialize BSP_AUDIO_IN MSP.

**Return values:**

None

Definition at line **1656** of file *stm32f769i_discovery_audio.c*.

References **AudioIn_Device**, **DFSDMx_ChannelMspDeInit()**, **DFSDMx_FilterMspDeInit()**, **haudio_in_sai**, **INPUT_DEVICE_DIGITAL_MIC**, and **SAI_AUDIO_IN_MspDeInit()**.

Referenced by **BSP_AUDIO_IN_DeInit()**.

```c
void BSP_AUDIO_IN_MspInit ( void )
```

Initialize BSP_AUDIO_IN MSP.

**Return values:**

None

Definition at line **1637** of file *stm32f769i_discovery_audio.c*.

References **AudioIn_Device**, **DFSDMx_ChannelMspInit()**, **DFSDMx_FilterMspInit()**, **haudio_in_sai**, **INPUT_DEVICE_DIGITAL_MIC**, and **SAI_AUDIO_IN_MspInit()**.

Referenced by **BSP_AUDIO_IN_InitEx()**.
uint8_t BSP_AUDIO_IN_Pause ( void )

Pause the audio file stream.

**Return values:**
- **AUDIO_OK** if correct communication, else wrong communication

Definition at line 1338 of file stm32f769i_discovery_audio.c.

References AUDIO_ERROR, AUDIO_OK, AudioIn_ChannelNumber, hAudioInBottomLeftFilter, hAudioInBottomRightFilter, hAudioInTopLeftFilter, and hAudioInTopRightFilter.

uint8_t BSP_AUDIO_IN_Record ( uint16_t * pbuf, uint32_t size )

Start audio recording.

**Parameters:**
- **pbuf:** Main buffer pointer for the recorded data storing
- **size:** Current size of the recorded buffer

**Return values:**
- **AUDIO_OK** if correct communication, else wrong communication

Definition at line 1224 of file stm32f769i_discovery_audio.c.

References AppBuffHalf, AppBuffTrigger, AUDIO_ERROR, AUDIO_OK, AudioIn_ChannelNumber, AudioIn_Device, haudio_in_sai, hAudioInBottomLeftFilter, hAudioInBottomRightFilter, hAudioInTopLeftFilter, hAudioInTopRightFilter, INPUT_DEVICE_DIGITAL_MIC,
**uint8_t BSP_AUDIO_IN_Resume ( void )**

Resume the audio file stream.

**Return values:**
- **AUDIO_OK** if correct communication, else wrong communication

Definition at line 1374 of file `stm32f769i_discovery_audio.c`.

References **AUDIO_ERROR**, **AUDIO_OK**, `AudioIn_ChannelNumber`, `hAudioInButtomLeftFilter`, `hAudioInButtomRightFilter`, `hAudioInTopLeftFilter`, `hAudioInTopRightFilter`, `pScratchBuff`, and **ScratchSize**.

**uint8_t BSP_AUDIO_IN_Stop ( void )**

Stop audio recording.

**Return values:**
- **AUDIO_OK** if correct communication, else wrong communication

Definition at line 1277 of file `stm32f769i_discovery_audio.c`.

References **AppBuffHalf**, **AppBuffTrigger**, `audio_drv`, **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, `AudioIn_ChannelNumber`, `AudioIn_Device`, `haudio_in_sai`, `hAudioInButtomLeftFilter`, `hAudioInButtomRightFilter`, `hAudioInTopLeftFilter`, `hAudioInTopRightFilter`, and **INPUT_DEVICE_DIGITAL_MIC**.
void BSP_AUDIO_IN_TransferComplete_CallBack ( void )

User callback when record buffer is filled.

**Return values:**

None

Definition at line 1605 of file stm32f769i_discovery_audio.c.

Referenced by HAL_DFSDM_FilterRegConvCpltCallback(),
HAL_DFSDM_FilterRegConvHalfCpltCallback(), and
HAL_SAI_RxCpltCallback().
# STM32F769I-Discovery BSP User Manual

## Directories

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### STM32F769I_DISCOVERY_AUDIO_Out Private Functions

STM32F769I_DISCOVERY AUDIO

### STM32F769I_DISCOVERY AUDIO
Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td><code>BSP_AUDIO_OUT_Init</code> (uint16_t OutputDevice, uint8_t Volume, uint32_t AudioFreq)</td>
<td>Configures the audio peripherals.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><code>BSP_AUDIO_OUT_Play</code> (uint16_t *pBuffer, uint32_t Size)</td>
<td>Starts playing audio stream from a data buffer for a determined size.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_AUDIO_OUT.ChangeBuffer</code> (uint16_t *pData, uint16_t Size)</td>
<td>Sends n-Bytes on the SAI interface.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><code>BSP_AUDIO_OUT_Pause</code> (void)</td>
<td>This function Pauses the audio file stream.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><code>BSP_AUDIO_OUT_Resume</code> (void)</td>
<td>Resumes the audio file stream.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><code>BSP_AUDIO_OUT_Stop</code> (uint32_t Option)</td>
<td>Stops audio playing and Power down the Audio Codec.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><code>BSP_AUDIO_OUT_SetVolume</code> (uint8_t Volume)</td>
<td>Controls the current audio volume level.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><code>BSP_AUDIO_OUT_SetMute</code> (uint32_t Cmd)</td>
<td>Enables or disables the MUTE mode by software.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><code>BSP_AUDIO_OUT_SetOutputMode</code> (uint8_t Output)</td>
<td>Switch dynamically (while audio file is played) the output target (speaker or headphone).</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_AUDIO_OUT_SetFrequency</code> (uint32_t AudioFreq)</td>
<td>Updates the audio frequency.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_AUDIO_OUT_SetAudioFrameSlot</code> (uint32_t AudioFrameSlot)</td>
<td>Updates the Audio frame slot configuration.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_AUDIO_OUT_DeInit</code> (void)</td>
<td>De-initializes the audio out peripheral.</td>
</tr>
<tr>
<td>Function Name</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>*<em>void HAL_SAI_TxCpltCallback (SAI_HandleTypeDef <em>hsai)</em></em></td>
<td>Tx Transfer completed callbacks.</td>
<td></td>
</tr>
<tr>
<td>*<em>void HAL_SAI_TxHalfCpltCallback (SAI_HandleTypeDef <em>hsai)</em></em></td>
<td>Tx Half Transfer completed callbacks.</td>
<td></td>
</tr>
<tr>
<td>*<em>void HAL_SAI_ErrorCallback (SAI_HandleTypeDef <em>hsai)</em></em></td>
<td>SAI error callbacks.</td>
<td></td>
</tr>
<tr>
<td><strong>__weak void BSP_AUDIO_OUT_TransferComplete_CallBack (void)</strong></td>
<td>Manages the DMA full Transfer complete event.</td>
<td></td>
</tr>
<tr>
<td><strong>__weak void BSP_AUDIO_OUT_HalfTransfer_CallBack (void)</strong></td>
<td>Manages the DMA Half Transfer complete event.</td>
<td></td>
</tr>
<tr>
<td><strong>__weak void BSP_AUDIO_OUT_Error_CallBack (void)</strong></td>
<td>Manages the DMA FIFO error event.</td>
<td></td>
</tr>
<tr>
<td>**__weak void BSP_AUDIO_OUT_MspInit (SAI_HandleTypeDef <em>hsai, void <em>Params)</em></em></td>
<td>Initializes BSP_AUDIO_OUT MSP.</td>
<td></td>
</tr>
<tr>
<td>**__weak void BSP_AUDIO_OUT_MspDeInit (SAI_HandleTypeDef <em>hsai, void <em>Params)</em></em></td>
<td>Deinitializes SAI MSP.</td>
<td></td>
</tr>
<tr>
<td>**__weak void BSP_AUDIO_OUT_ClockConfig (SAI_HandleTypeDef <em>hsai, uint32_t AudioFreq, void <em>Params)</em></em></td>
<td>Clock Config.</td>
<td></td>
</tr>
<tr>
<td>**static void SAI_AUDIO_IN_MspInit (SAI_HandleTypeDef <em>hsai, void <em>Params)</em></em></td>
<td>Initializes SAI Audio IN MSP.</td>
<td></td>
</tr>
<tr>
<td>**static void SAI_AUDIO_IN_MspDeInit (SAI_HandleTypeDef <em>hsai, void <em>Params)</em></em></td>
<td>De-Initializes SAI Audio IN MSP.</td>
<td></td>
</tr>
<tr>
<td><strong>static void SAIx_Out_Init (uint32_t AudioFreq)</strong></td>
<td>Initializes the Audio Codec audio interface (SAI).</td>
<td></td>
</tr>
<tr>
<td><strong>static void SAIx_Out_DeInit (void)</strong></td>
<td>Deinitializes the Audio Codec audio interface (SAI).</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>static void SAIx_In_Init (uint32_t AudioFreq)</td>
<td>Initializes the Audio Codec audio interface (SAI).</td>
<td></td>
</tr>
<tr>
<td>static void SAIx_In_DeInit (void)</td>
<td>Deinitializes the output Audio Codec audio interface (SAI).</td>
<td></td>
</tr>
</tbody>
</table>
Function Documentation

```c
void BSP_AUDIO_OUT_ChangeBuffer ( uint16_t * pData, uint16_t Size )
```

Sends n-Bytes on the SAI interface.

**Parameters:**
- **pData:** pointer on data address
- **Size:** number of data to be written

**Return values:**
- **None**

Definition at line 369 of file `stm32f769i_discovery_audio.c`.

References `haudio_out_sai`.

```c
__weak void BSP_AUDIO_OUT_ClockConfig ( SAI_HandleTypeDef * hsai, uint32_t AudioFreq, void * Params )
```

Clock Config.

**Parameters:**
- **hsai:** might be required to set audio peripheral prescaler if any.
- **AudioFreq:** Audio frequency used to play the audio stream.
- **Params**

**Note:**
This API is called by `BSP_AUDIO_OUT_Init()` and `BSP_AUDIO_OUT_SetFrequency()` Being __weak it can be overwritten by the application

**Return values:**

None

Definition at line 851 of file `stm32f769i_discovery_audio.c`.

Referenced by `BSP_AUDIO_IN_InitEx()`, `BSP_AUDIO_OUT_Init()`, and `BSP_AUDIO_OUT_SetFrequency()`.

```c
void BSP_AUDIO_OUT_DeInit ( void )
```

De-initializes the audio out peripheral.

**Return values:**

None

Definition at line 562 of file `stm32f769i_discovery_audio.c`.

References `BSP_AUDIO_OUT_MspDeInit()`, `haudio_out_sai`, and `SAIx_Out_DeInit()`.

```c
__weak void BSP_AUDIO_OUT_Error_CallBack ( void )
```

Manages the DMA FIFO error event.

**Return values:**

None

Definition at line 630 of file `stm32f769i_discovery_audio.c`.

Referenced by `HAL_SAI_ErrorCallback()`.
__weak void BSP_AUDIO_OUT_HalfTransfer_CallBack ( void )

Manages the DMA Half Transfer complete event.

**Return values:**

None

Definition at line 622 of file stm32f769i_discovery_audio.c.

Referenced by HAL_SAI_TxHalfCpltCallback().

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

Configures the audio peripherals.

**Parameters:**

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

**Return values:**

- **AUDIO_OK** if correct communication, else wrong communication

Definition at line 295 of file stm32f769i_discovery_audio.c.

References audio_drv, AUDIO_ERROR, AUDIO_I2C_ADDRESS, AUDIO_OK, AUDIO_OUT_SAIx, BSP_AUDIO_OUT_ClockConfig().
BSP_AUDIO_OUT_MspInit(), haudio_out_sai, SAIx_Out_DeInit(), and SAIx_Out_Init().

```c
__weak void BSP_AUDIO_OUT_MspDeInit ( SAI_HandleTypeDef * hsa, void * Params )
```

Deinitializes SAI MSP.

**Parameters:**
- `hsai`: SAI handle
- `Params`

**Return values:**
- None

Definition at line 812 of file `stm32f769i_discovery_audio.c`.

References AUDIO_OUT_SAIx, AUDIO_OUT_SAIx_CLK_DISABLE, AUDIO_OUT_SAIx_DMAx_IRQ, AUDIO_OUT_SAIx_FS_PIN, AUDIO_OUT_SAIx_MCLK_GPIO_PORT, AUDIO_OUT_SAIx_MCLK_PIN, AUDIO_OUT_SAIx_SCK_PIN, AUDIO_OUT_SAIx_SD_FS_SCK_GPIO_PORT, and AUDIO_OUT_SAIx_SD_PIN.

Referenced by BSP_AUDIO_OUT_DeInit().

```c
__weak void BSP_AUDIO_OUT_MspInit ( SAI_HandleTypeDef * hsa, void * Params )
```

Initializes BSP_AUDIO_OUT MSP.

**Parameters:**
hsai: SAI handle
Params

Return values:
None

Definition at line 640 of file stm32f769i_discovery_audio.c.

References AUDIO_OUT_IRQPREPRIO, AUDIO_OUT_SAIx, AUDIO_OUT_SAIx_AF, AUDIO_OUT_SAIx_CLK_ENABLE, AUDIO_OUT_SAIx_DMAx_CHANNEL, AUDIO_OUT_SAIx_DMAx_CLK_ENABLE, AUDIO_OUT_SAIx_DMAx_IRQ, AUDIO_OUT_SAIx_DMAx_MEM_DATA_SIZE, AUDIO_OUT_SAIx_DMAx_PERIPH_DATA_SIZE, AUDIO_OUT_SAIx_DMAx_STREAM, AUDIO_OUT_SAIx_FS_PIN, AUDIO_OUT_SAIx_MCLK_ENABLE, AUDIO_OUT_SAIx_MCLK_GPIO_PORT, AUDIO_OUT_SAIx_MCLK_PIN, AUDIO_OUT_SAIx_SCK_PIN, AUDIO_OUT_SAIx_SD_FS_CLK_ENABLE, AUDIO_OUT_SAIx_SD_FS_SCK_GPIO_PORT, and AUDIO_OUT_SAIx_SD_PIN.

Referenced by BSP_AUDIO_IN_InitEx(), and BSP_AUDIO_OUT_Init().

uint8_t BSP_AUDIO_OUT_Pause ( void )

This function Pauses the audio file stream.

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

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

**Return values:**

- **AUDIO_OK** if correct communication, else wrong communication

Definition at line 382 of file `stm32f769i_discovery_audio.c`.

References `audio_drv`, `AUDIO_ERROR`, `AUDIO_I2C_ADDRESS`, `AUDIO_OK`, and `haudio_out_sai`.

```c
uint8_t BSP_AUDIO_OUT_Play (uint16_t * pBuffer,
                           uint32_t Size)
```

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

**Parameters:**

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

**Return values:**

- **AUDIO_OK** if correct communication, else wrong communication

Definition at line 347 of file `stm32f769i_discovery_audio.c`.

References `audio_drv`, `AUDIO_ERROR`, `AUDIO_I2C_ADDRESS`, `AUDIO_OK`, `AUDIODATA_SIZE`, `DMA_MAX`, and `haudio_out_sai`.

```c
uint8_t BSP_AUDIO_OUT_Resume (void )
```

Resumes the audio file stream.

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

**Return values:**

- `AUDIO_OK` if correct communication, else wrong communication

Definition at line 406 of file `stm32f769i_discovery_audio.c`.

References `audio_drv`, `AUDIO_ERROR`, `AUDIO_I2C_ADDRESS`, `AUDIO_OK`, and `haudio_out_sai`.

```c
void BSP_AUDIO_OUT_SetAudioFrameSlot ( uint32_t AudioFrameSlot )
```

Updates the Audio frame slot configuration.

**Parameters:**

- `AudioFrameSlot` specifies the audio Frame slot

**Note:**

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

**Return values:**

- `None`

Definition at line 545 of file `stm32f769i_discovery_audio.c`.

References `haudio_out_sai`.

```c
void BSP_AUDIO_OUT_SetFrequency ( uint32_t AudioFreq )
```

Updates the audio frequency.
Parameters:

- **AudioFreq**: Audio frequency used to play the audio stream.

Note:

This API should be called after the `BSP_AUDIO_OUT_Init()` to adjust the audio frequency.

Return values:

None

Definition at line 522 of file `stm32f769i_discovery_audio.c`.

References `BSP_AUDIO_OUT_CLOCK_CONFIG()`, and `haudio_out_sai`.

```c
uint8_t BSP_AUDIO_OUT_SetMute ( uint32_t Cmd )
```

Enables or disables the MUTE mode by software.

Parameters:

- **Cmd**: Could be `AUDIO_MUTE_ON` to mute sound or `AUDIO_MUTE_OFF` to unmute the codec and restore previous volume level.

Return values:

- **AUDIO_OK** if correct communication, else wrong communication

Definition at line 480 of file `stm32f769i_discovery_audio.c`.

References `audio_drv`, `AUDIO_ERROR`, `AUDIO_I2C_ADDRESS`, and `AUDIO_OK`.

```c
uint8_t BSP_AUDIO_OUT_SetOutputMode ( uint8_t Output )
```

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

**Parameters:**

**Output:** The audio output target:
- OUTPUT_DEVICE_SPEAKER,
- OUTPUT_DEVICE_HEADPHONE or
- OUTPUT_DEVICE_BOTH

**Return values:**

**AUDIO_OK** if correct communication, else wrong communication

Definition at line 501 of file stm32f769i_discovery_audio.c.

References audio_drv, AUDIO_ERROR, AUDIO_I2C_ADDRESS, and AUDIO_OK.

```c
uint8_t BSP_AUDIO_OUT_SetVolume ( uint8_t Volume )
```

Controls the current audio volume level.

**Parameters:**

**Volume:** Volume level to be set in percentage from 0% to 100% (0 for Mute and 100 for Max volume level).

**Return values:**

**AUDIO_OK** if correct communication, else wrong communication

Definition at line 460 of file stm32f769i_discovery_audio.c.

References audio_drv, AUDIO_ERROR, AUDIO_I2C_ADDRESS, and AUDIO_OK.

```c
uint8_t BSP_AUDIO_OUT_Stop ( uint32_t Option )
```
Stops audio playing and Power down the Audio Codec.

**Parameters:**

- **Option:** could be one of the following parameters
  - **CODEC_PDWN_SW:** for software power off (by writing registers). Then no need to reconfigure the Codec after power on.
  - **CODEC_PDWN_HW:** completely shut down the codec (physically). Then need to reconfigure the Codec after power on.

**Return values:**

- **AUDIO_OK** if correct communication, else wrong communication

Definition at line 432 of file `stm32f769i_discovery_audio.c`.

References `audio_drv`, `AUDIO_ERROR`, `AUDIO_I2C_ADDRESS`, `AUDIO_OK`, and `haudio_out_sai`.

```c
__weak void BSP_AUDIO_OUT_TransferComplete_CallBack ( void 

Manages the DMA full Transfer complete event.

**Return values:**

- **None**

Definition at line 614 of file `stm32f769i_discovery_audio.c`.

Referenced by `HAL_SAI_TxCpltCallback()`.

```c
void HAL_SAI_ErrorCallback ( SAI_HandleTypeDef * hsai )
```

SAI error callbacks.
**Parameters:**
- `hsai`: SAI handle

**Return values:**
- `None`

Definition at line 598 of file `stm32f769i_discovery_audio.c`.

References `AUDIO_OUT_SAIx`, `BSP_AUDIO_IN_Error_CallBack()`, and `BSP_AUDIO_OUT_Error_CallBack()`.

```c
void HAL_SAI_TxCpltCallback ( SAI_HandleTypeDef * hsai )
```

Tx Transfer completed callbacks.

**Parameters:**
- `hsai`: SAI handle

**Return values:**
- `None`

Definition at line 574 of file `stm32f769i_discovery_audio.c`.

References `BSP_AUDIO_OUT_TransferComplete_CallBack()`.

```c
void HAL_SAI_TxHalfCpltCallback ( SAI_HandleTypeDef * hsai )
```

Tx Half Transfer completed callbacks.

**Parameters:**
- `hsai`: SAI handle

**Return values:**
- `None`
Definition at line 586 of file stm32f769i_discovery_audio.c.

References BSP_AUDIO_OUT_HalfTransfer_CallBack().

```c
static void SAI_AUDIO_IN_MspDeInit ( SAI_HandleTypeDef * hsai,
                                      void * Params)
```

De-Initializes SAI Audio IN MSP.

**Parameters:**
- `hsai`: SAI handle
- `Params`

**Return values:**
- `None`

Definition at line 779 of file stm32f769i_discovery_audio.c.

References AUDIO_IN_INT_GPIO_PIN,
AUDIO_IN_INT_GPIO_PORT, AUDIO_IN_SAIx,
AUDIO_IN_SAIx_CLK_DISABLE, AUDIO_IN_SAIx_DMAx_IRQ,
AUDIO_IN_SAIx_SD_GPIO_PORT, and AUDIO_IN_SAIx_SD_PIN.

Referenced by BSP_AUDIO_IN_MspDeInit().

```c
static void SAI_AUDIO_IN_MspInit ( SAI_HandleTypeDef * hsai,
                                     void * Params)
```

Initializes SAI Audio IN MSP.

**Parameters:**
- `hsai`: SAI handle
static void SAIx_In_DeInit ( void ) [static]

Deinitializes the output Audio Codec audio interface (SAI).

Return values:

None

Definition at line 1090 of file stm32f769i_discovery_audio.c.

References AUDIO_IN_SAIx, AUDIO_OUT_SAIx, haudio_in_sai, and haudio_out_sai.

Referenced by BSP_AUDIO_IN_DeInit(), and BSP_AUDIO_IN_InitEx().
static void SAIx_In_Init (uint32_t AudioFreq) [static]

Initializes the Audio Codec audio interface (SAI).

**Parameters:**

- **AudioFreq:** Audio frequency to be configured for the SAI peripheral.

**Note:**

The default SlotActive configuration is set to CODEC_AUDIOFRAME SLOT_0123 and user can update this configuration using

**Return values:**

None

Definition at line 977 of file stm32f769i_discovery_audio.c.

References AUDIO_IN_SAIx, AUDIO_OUT_SAIx, CODEC_AUDIOFRAME SLOT_02, haudio_in_sai, and haudio_out_sai.

Referenced by BSP_AUDIO_IN_InitEx().

---

static void SAIx_Out_DeInit ( void ) [static]

Deinitializes the Audio Codec audio interface (SAI).

**Return values:**

None

Definition at line 959 of file stm32f769i_discovery_audio.c.

References AUDIO_OUT_SAIx, and haudio_out_sai.

Referenced by BSP_AUDIO_OUT_DeInit(), and BSP_AUDIO_OUT_Init().
static void SAIx_Out_Init ( uint32_t AudioFreq ) [static]

Initializes the Audio Codec audio interface (SAI).

**Parameters:**

AudioFreq: Audio frequency to be configured for the SAI peripheral.

**Note:**

The default SlotActive configuration is set to CODEC_AUDIOFRAME_SLOT_0123 and user can update this configuration using

**Return values:**

None

Definition at line 900 of file stm32f769i_discovery_audio.c.

References AUDIO_OUT_SAIx, CODEC_AUDIOFRAME_SLOT_0123, and haudio_out_sai.

Referenced by BSP_AUDIO_OUT_Init().
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**STM32F769I_DISCOVERY_AUDIO_OUT**

Exported Functions

**STM32F769I_DISCOVERY AUDIO**
## Functions

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Function Name</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td>BSP_AUDIO_OUT_Init</td>
<td>(uint16_t OutputDevice, uint8_t Volume, uint32_t AudioFreq)</td>
<td>Configures the audio peripherals.</td>
</tr>
<tr>
<td>void</td>
<td>BSP_AUDIO_OUT_DeInit</td>
<td>(void)</td>
<td>De-initializes the audio out peripheral.</td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_AUDIO_OUT_Play</td>
<td>(uint16_t *pBuffer, uint32_t Size)</td>
<td>Starts playing audio stream from a data buffer for a determined size.</td>
</tr>
<tr>
<td>void</td>
<td>BSP_AUDIO_OUT_ChangeBuffer</td>
<td>(uint16_t *pData, uint16_t Size)</td>
<td>Sends n-Bytes on the SAI interface.</td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_AUDIO_OUT_Pause</td>
<td>(void)</td>
<td>This function Pauses the audio file stream.</td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_AUDIO_OUT_Resume</td>
<td>(void)</td>
<td>Resumes the audio file stream.</td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_AUDIO_OUT_Stop</td>
<td>(uint32_t Option)</td>
<td>Stops audio playing and Power down the Audio Codec.</td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_AUDIO_OUT_SetVolume</td>
<td>(uint8_t Volume)</td>
<td>Controls the current audio volume level.</td>
</tr>
<tr>
<td>void</td>
<td>BSP_AUDIO_OUT_SetFrequency</td>
<td>(uint32_t AudioFreq)</td>
<td>Updates the audio frequency.</td>
</tr>
<tr>
<td>void</td>
<td>BSP_AUDIO_OUT_SetAudioFrameSlot</td>
<td>(uint32_t AudioFrameSlot)</td>
<td>Updates the Audio frame slot configuration.</td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_AUDIO_OUT_SetMute</td>
<td>(uint32_t Cmd)</td>
<td>Enables or disables the MUTE mode by software.</td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_AUDIO_OUT_SetOutputMode</td>
<td>(uint8_t Output)</td>
<td>Switch dynamically (while audio file is played) the output target (speaker or headphone).</td>
</tr>
<tr>
<td>void</td>
<td>BSP_AUDIO_OUT_TransferComplete_CallBack</td>
<td>(void)</td>
<td>Manages the DMA full Transfer complete event.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>BSP_AUDIO_OUT_HalfTransfer_CallBack (void)</code></td>
<td>Manages the DMA Half Transfer complete event.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>BSP_AUDIO_OUT_Error_CallBack (void)</code></td>
<td>Manages the DMA FIFO error event.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>BSP_AUDIO_OUT_ClockConfig</code></td>
<td>Clock Config.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>BSP_AUDIO_OUT_MspInit</code></td>
<td>Initializes BSP_AUDIO_OUT MSP.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>BSP_AUDIO_OUT_MspDeInit</code></td>
<td>Deinitializes SAI MSP.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Function Documentation**

```c
void BSP_AUDIO_OUT_ChangeBuffer ( uint16_t * pData, uint16_t Size )
```

Sends n-Bytes on the SAI interface.

**Parameters:**
- **pData:** pointer on data address
- **Size:** number of data to be written

**Return values:**
- **None**

Definition at line 369 of file stm32f769i_discovery_audio.c.

References haudio_out_sai.

```c
void BSP_AUDIO_OUT_ClockConfig ( SAI_HandleTypeDef * hsai, uint32_t AudioFreq, void * Params )
```

Clock Config.

**Parameters:**
- **hsai:** might be required to set audio peripheral predivider if any.
- **AudioFreq:** Audio frequency used to play the audio stream.
- **Params**

**Note:**
This API is called by `BSP_AUDIO_OUT_Init()` and `BSP_AUDIO_OUT_SetFrequency()` Being __weak it can be overwritten by the application.

**Return values:**

None

Definition at line 851 of file stm32f769i_discovery_audio.c.

Referenced by `BSP_AUDIO_IN_InitEx()`, `BSP_AUDIO_OUT_Init()`, and `BSP_AUDIO_OUT_SetFrequency()`.

```c
void BSP_AUDIO_OUT_DeInit ( void )
```

De-initializes the audio out peripheral.

**Return values:**

None

Definition at line 562 of file stm32f769i_discovery_audio.c.

References `BSP_AUDIO_OUT_MspDeInit()`, `haudio_out_sai`, and `SAIx_Out_DeInit()`.

```c
void BSP_AUDIO_OUT_Error_CallBack ( void )
```

Manages the DMA FIFO error event.

**Return values:**

None

Definition at line 630 of file stm32f769i_discovery_audio.c.

Referenced by `HAL_SAI_ErrorCallback()`.
void BSP_AUDIO_OUT_HalfTransfer_CallBack ( void )

Manages the DMA Half Transfer complete event.

**Return values:**
- *None*

Definition at line 622 of file `stm32f769i_discovery_audio.c`.

Referenced by `HAL_SAI_TxHalfCpltCallback()`.

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

Configures the audio peripherals.

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

**Return values:**
- **AUDIO_OK** if correct communication, else wrong communication

Definition at line 295 of file `stm32f769i_discovery_audio.c`.

References `audio_drv`, `AUDIO_ERROR`, `AUDIO_I2C_ADDRESS`, `AUDIO_OK`, `AUDIO_OUT_SAIx`, `BSP_AUDIO_OUT_ClockConfig()`,
BSP_AUDIO_OUT_MspInit(), haudio_out_sai, SAIx_Out_DeInit(), and SAIx_Out_Init().

```c
void BSP_AUDIO_OUT_MspDeInit ( SAI_HandleTypeDef * hsai, void * Params )
```

Deinitializes SAI MSP.

**Parameters:**
- **hsai:** SAI handle
- **Params**

**Return values:**
- None

Definition at line 812 of file stm32f769i_discovery_audio.c.

References AUDIO_OUT_SAIx, AUDIO_OUT_SAIx_CLK_DISABLE, AUDIO_OUT_SAIx_DMAx_IRQ, AUDIO_OUT_SAIx_FS_PIN, AUDIO_OUT_SAIx_MCLK_GPIO_PORT, AUDIO_OUT_SAIx_MCLK_PIN, AUDIO_OUT_SAIx_SCK_PIN, AUDIO_OUT_SAIx_SD_FS_SCK_GPIO_PORT, and AUDIO_OUT_SAIx_SD_PIN.

Referenced by BSP_AUDIO_OUT_DeInit().

```c
void BSP_AUDIO_OUT_MspInit ( SAI_HandleTypeDef * hsai, void * Params )
```

Initializes BSP_AUDIO_OUT MSP.

**Parameters:**
**hsai:** SAI handle

**Params**

**Return values:**

**None**

Definition at line 640 of file `stm32f769i_discovery_audio.c`.

References `AUDIO_OUT_IRQ_PREPRIO`, `AUDIO_OUT_SAIx`, `AUDIO_OUT_SAIx_AF`, `AUDIO_OUT_SAIx_CLK_ENABLE`, `AUDIO_OUT_SAIx_DMAx_CHANNEL`, `AUDIO_OUT_SAIx_DMAx_CLK_ENABLE`, `AUDIO_OUT_SAIx_DMAx_IRQ`, `AUDIO_OUT_SAIx_DMAx_MEM_DATA_SIZE`, `AUDIO_OUT_SAIx_DMAx_PERIPH_DATA_SIZE`, `AUDIO_OUT_SAIx_DMAx_STREAM`, `AUDIO_OUT_SAIx_FS_PIN`, `AUDIO_OUT_SAIx_MCLK_ENABLE`, `AUDIO_OUT_SAIx_MCLK_GPIO_PORT`, `AUDIO_OUT_SAIx_MCLK_PIN`, `AUDIO_OUT_SAIx_SCK_PIN`, `AUDIO_OUT_SAIx_SD_FS_CLK_ENABLE`, `AUDIO_OUT_SAIx_SD_FS_SCK_GPIO_PORT`, and `AUDIO_OUT_SAIx_SD_PIN`.

Referenced by `BSP_AUDIO_IN_InitEx()`, and `BSP_AUDIO_OUT_Init()`.

```c
uint8_t BSP_AUDIO_OUT_Pause ( void )
```

This function Pauses the audio file stream.

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

**Note:**

When calling `BSP_AUDIO_OUT_Pause()` function for pause, only `BSP_AUDIO_OUT_Resume()` function should be called for resume (use of `BSP_AUDIO_OUT_Play()` function for resume
could lead to unexpected behaviour).

**Return values:**

- **AUDIO_OK**  if correct communication, else wrong communication

Definition at line 382 of file `stm32f769i_discovery_audio.c`.

References `audio_drv`, `AUDIO_ERROR`, `AUDIO_I2C_ADDRESS`, `AUDIO_OK`, and `haudio_out_sai`.

```c
uint8_t BSP_AUDIO_OUT_Play ( uint16_t * pBuffer,
                         uint32_t Size )
```

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

**Parameters:**

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

**Return values:**

- **AUDIO_OK**  if correct communication, else wrong communication

Definition at line 347 of file `stm32f769i_discovery_audio.c`.

References `audio_drv`, `AUDIO_ERROR`, `AUDIO_I2C_ADDRESS`, `AUDIO_OK`, `AUDIODATA_SIZE`, `DMA_MAX`, and `haudio_out_sai`.

```c
uint8_t BSP_AUDIO_OUT_Resume ( void )
```

Resumes the audio file stream.

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

**Return values:**

- `AUDIO_OK` if correct communication, else wrong communication

Definition at line 406 of file `stm32f769i_discovery_audio.c`.

References `audio_drv`, `AUDIO_ERROR`, `AUDIO_I2C_ADDRESS`, `AUDIO_OK`, and `haudio_out_sai`.

```c
void BSP_AUDIO_OUT_SetAudioFrameSlot ( uint32_t AudioFrameSlot );
```

Updates the Audio frame slot configuration.

**Parameters:**

- `AudioFrameSlot`: specifies the audio Frame slot

**Note:**

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

**Return values:**

- None

Definition at line 545 of file `stm32f769i_discovery_audio.c`.

References `haudio_out_sai`.

```c
void BSP_AUDIO_OUT_SetFrequency ( uint32_t AudioFreq )
```

Updates the audio frequency.
Parameters:

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

Note:

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

Return values:

**None**

Definition at line 522 of file **stm32f769i_discovery_audio.c**.

References **BSP_AUDIO_OUT_ClockConfig()**, and **haudio_out_sai**.

```c
uint8_t BSP_AUDIO_OUT_SetMute ( uint32_t Cmd )
```

Enables or disables the MUTE mode by software.

Parameters:

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

Return values:

**AUDIO_OK** if correct communication, else wrong communication

Definition at line 480 of file **stm32f769i_discovery_audio.c**.

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

```c
uint8_t BSP_AUDIO_OUT_SetOutputMode ( uint8_t Output )
```

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

**Parameters:**

- **Output:** The audio output target:
  - OUTPUT_DEVICE_SPEAKER,
  - OUTPUT_DEVICE_HEADPHONE or
  - OUTPUT_DEVICE_BOTH

**Return values:**

- **AUDIO_OK** if correct communication, else wrong communication

Definition at line 501 of file `stm32f769i_discovery_audio.c`.

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

```c
uint8_t BSP_AUDIO_OUT_SetVolume ( uint8_t Volume )
```

Controls the current audio volume level.

**Parameters:**

- **Volume:** Volume level to be set in percentage from 0% to 100% (0 for Mute and 100 for Max volume level).

**Return values:**

- **AUDIO_OK** if correct communication, else wrong communication

Definition at line 460 of file `stm32f769i_discovery_audio.c`.

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

```c
uint8_t BSP_AUDIO_OUT_Stop ( uint32_t Option )
```
Stops audio playing and Power down the Audio Codec.

**Parameters:**

**Option:** could be one of the following parameters

- **CODEC_PDWN_SW:** for software power off (by writing registers). Then no need to reconfigure the Codec after power on.
- **CODEC_PDWN_HW:** completely shut down the codec (physically). Then need to reconfigure the Codec after power on.

**Return values:**

**AUDIO_OK** if correct communication, else wrong communication

Definition at line 432 of file `stm32f769i_discovery_audio.c`.

References `audio_drv`, `AUDIO_ERROR`, `AUDIO_I2C_ADDRESS`, `AUDIO_OK`, and `haudio_out_sai`.

```c
void BSP_AUDIO_OUT_TransferComplete_CallBack ( void )
```

Manages the DMA full Transfer complete event.

**Return values:**

**None**

Definition at line 614 of file `stm32f769i_discovery_audio.c`.

Referenced by `HAL_SAI_TxCpltCallback()`.
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
<th>Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Directory</strong></td>
</tr>
</tbody>
</table>

**EEPROM Private Functions**

STM32F769I_DISCOVERY EEPROM
## Functions

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

```c
uint8_t BSP_EEPROM_DeInit ( void )
```

DeInitializes the EEPROM.

**Return values:**

- **EEPROM** state

Definition at line 177 of file `stm32f769i_discovery_eeprom.c`.

References **EEPROM_OK**.

```c
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 154 of file `stm32f769i_discovery_eeprom.c`.

References **EEPROM_FAIL**, **EEPROM_I2C_ADDRESS_A01**, **EEPROM_I2C_ADDRESS_A02**, **EEPROM_IO_Init()**, **EEPROM_IO_IsDeviceReady()**, **EEPROM_MAX_TRIALS**, **EEPROM_OK**, and **EEPROMAddress**.

```c
uint32_t BSP_EEPROM_ReadBuffer ( uint8_t * pBuffer,
```
Reads a block of data from the EEPROM.

**Parameters:**
- `pBuffer`: pointer to the buffer that receives the data read from the EEPROM.
- `ReadAddr`: EEPROM’s internal address to start reading from.
- `NumByteToRead`: pointer to the variable holding number of bytes to be read from the EEPROM.

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

**Return values:**
- `EEPROM_OK` (0) if operation is correctly performed, else return value different from `EEPROM_OK` (0) or the timeout user callback.

Definition at line 198 of file `stm32f769i_discovery_eeprom.c`.

References `BSP_EEPROM_TIMEOUT_UserCallback()`, `EEPROM_FAIL`, `EEPROM_IO_ReadData()`, `EEPROM_OK`, `EEPROMAddress`, and `EEPROMDataRead`.

```c
__weak void BSP_EEPROM_TIMEOUT_UserCallback ( void )
```

Basic management of the timeout situation.

**Return values:**
uint32_t BSP_EEPROM_WaitEepromStandbyState ( void )

Wait for EEPROM Standby state.

**Note:**
This function allows to wait and check that EEPROM has finished the last operation. It is mostly used after Write operation: after receiving the buffer to be written, the EEPROM may need additional time to actually perform the write operation. During this time, it doesn't answer to I2C packets addressed to it. Once the write operation is complete the EEPROM responds to its address.

**Return values:**

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

Definition at line 437 of file stm32f769i_discovery_eeprom.c.

References BSP_EEPROM_TIMEOUT_UserCallback(), EEPROM_IO_IsDeviceReady(), EEPROM_MAX_TRIALS, EEPROM_OK, EEPROM_TIMEOUT, and EEPROMAddress.

Referenced by BSP_EEPROM_WritePage().

uint32_t BSP_EEPROM_WriteBuffer ( uint8_t * pBuffer, uint16_t WriteAddr,
writes buffer of data to the I2C EEPROM.

**Parameters:**

- **pBuffer:** pointer to the buffer containing the data to be written to the EEPROM.
- **WriteAddr:** EEPROM's internal address to write to.
- **NumByteToWrite:** number of bytes to write to the EEPROM.

**Return values:**

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

Definition at line 280 of file `stm32f769i_discovery_eeprom.c`.

References `BSP_EEPROM_WritePage()`, `EEPROM_OK`, and `EEPROM_PAGESIZE`.

```c
uint16_t NumByteToWrite
)

writes more than one byte to the EEPROM with a single WRITE cycle.

**Note:**

The number of bytes (combined to write start address) must not cross the EEPROM page boundary. This function can only write into the boundaries of an EEPROM page. This function doesn't check on boundaries condition (in this driver the function `BSP_EEPROM_WriteBuffer()` which calls
**BSP_EEPROM_WritePage()** is responsible of checking on Page boundaries).

**Parameters:**

- **pBuffer:** pointer to the buffer containing the data to be written to the EEPROM.
- **WriteAddr:** EEPROM's internal address to write to.
- **NumByteToWrite:** pointer to the variable holding number of bytes to be written into the EEPROM.

**Note:**

The variable pointed by NumByteToWrite is reset to 0 when all the data are written to the EEPROM. Application should monitor this variable in order know when the transfer is complete. This function just configure the communication and enable the DMA channel to transfer data. Meanwhile, the user application may perform other tasks in parallel.

**Return values:**

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

Definition at line 245 of file *stm32f769i_discovery_eeprom.c*.

References **BSP_EEPROM_TIMEOUT_UserCallback()**, **BSP_EEPROM_WaitEepromStandbyState()**, **EEPROM_FAIL**, **EEPROM_IO_WriteData()**, **EEPROM_OK**, **EEPROMAddress**, and **EEPROMDataWrite**.

Referenced by **BSP_EEPROM_WriteBuffer()**.
## EEPROM Exported Functions

**STM32F769I_DISCOVERY EEPROM**
### Functions

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint32_t</td>
<td>BSP_EEPROM_Init (void)</td>
<td>Initializes peripherals used by the I2C EEPROM driver.</td>
</tr>
<tr>
<td>uint8_t</td>
<td>BSP_EEPROM_DeInit (void)</td>
<td>DeInitializes the EEPROM.</td>
</tr>
<tr>
<td>uint32_t</td>
<td>BSP_EEPROM_ReadBuffer (uint8_t *pBuffer, uint16_t ReadAddr, uint16_t *NumByteToRead)</td>
<td>Reads a block of data from the EEPROM.</td>
</tr>
<tr>
<td>uint32_t</td>
<td>BSP_EEPROM_WritePage (uint8_t *pBuffer, uint16_t WriteAddr, uint8_t *NumByteToWrite)</td>
<td>Writes more than one byte to the EEPROM with a single WRITE cycle.</td>
</tr>
<tr>
<td>uint32_t</td>
<td>BSP_EEPROM_WriteBuffer (uint8_t *pBuffer, uint16_t WriteAddr, uint16_t NumByteToWrite)</td>
<td>Writes buffer of data to the I2C EEPROM.</td>
</tr>
<tr>
<td>uint32_t</td>
<td>BSP_EEPROM_WaitEepromStandbyState (void)</td>
<td>Wait for EEPROM Standby state.</td>
</tr>
<tr>
<td>void</td>
<td>BSP_EEPROM_TIMEOUT_UserCallback (void)</td>
<td>Basic management of the timeout situation.</td>
</tr>
<tr>
<td>void</td>
<td>EEPROM_IO_Init (void)</td>
<td>Initializes peripherals used by the I2C EEPROM driver.</td>
</tr>
<tr>
<td>HAL_StatusTypeDef</td>
<td>EEPROM_IO_WriteData (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize)</td>
<td>Write data to I2C EEPROM driver in using DMA channel.</td>
</tr>
<tr>
<td>HAL_StatusTypeDef</td>
<td>EEPROM_IO_ReadData (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize)</td>
<td></td>
</tr>
<tr>
<td>HAL_StatusTypeDef</td>
<td>EEPROM_IO_IsDeviceReady (uint16_t DevAddress, uint32_t Trials)</td>
<td>Checks if target device is ready for communication.</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>Read data from I2C EEPROM driver in using DMA channel.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Function Documentation

### `uint8_t BSP_EEPROM_DeInit ( void )`

DeInitializes the EEPROM.

**Return values:**
- **EEPROM** state

Definition at line 177 of file `stm32f769i_discovery_eeprom.c`.

References **EEPROM_OK**.

### `uint32_t BSP_EEPROM_Init ( void )`

Initializes peripherals used by the I2C EEPROM driver.

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

**Return values:**
- **EEPROM_OK** (0) if operation is correctly performed, else return value different from EEPROM_OK (0)

Definition at line 154 of file `stm32f769i_discovery_eeprom.c`.

References **EEPROM_FAIL**, **EEPROM_I2C_ADDRESS_A01**, **EEPROM_I2C_ADDRESS_A02**, **EEPROM_IO_Init()**, **EEPROM_IO_IsDeviceReady()**, **EEPROM_MAX_TRIALS**, **EEPROM_OK**, and **EEPROMAddress**.

### `uint32_t BSP_EEPROM_ReadBuffer ( uint8_t * pBuffer,`
uint16_t ReadAddr, uint16_t * NumByteToRead
)

Reads a block of data from the EEPROM.

Parameters:
  pBuffer,: pointer to the buffer that receives the data read from the EEPROM.
  ReadAddr,: EEPROM’s internal address to start reading from.
  NumByteToRead,: pointer to the variable holding number of bytes to be read from the EEPROM.

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

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

Definition at line 198 of file stm32f769i_discovery_eeprom.c.

References BSP_EEPROM_TIMEOUT_UserCallback(), EEPROM_FAIL, EEPROM_IO_ReadData(), EEPROM_OK, EEPROMAddress, and EEPROMDataRead.

void BSP_EEPROM_TIMEOUT_UserCallback ( void )

Basic management of the timeout situation.

Return values:
Definition at line 453 of file stm32f769i_discovery_eeprom.c.

Referenced by BSP_EEPROM_ReadBuffer(), 
BSP_EEPROM_WaitEepromStandbyState(), and 
BSP_EEPROM_WritePage().

```c
uint32_t BSP_EEPROM_WaitEepromStandbyState ( void )
```

Wait for EEPROM Standby state.

**Note:**
This function allows to wait and check that EEPROM has finished the last operation. It is mostly used after Write operation: after receiving the buffer to be written, the EEPROM may need additional time to actually perform the write operation. During this time, it doesn't answer to I2C packets addressed to it. Once the write operation is complete the EEPROM responds to its address.

**Return values:**

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

Definition at line 437 of file stm32f769i_discovery_eeprom.c.

References BSP_EEPROM_TIMEOUT_UserCallback(), 
EEPROM_IO_IsDeviceReady(), EEPROM_MAX_TRIALS, 
EEPROM_OK, EEPROM_TIMEOUT, and EEPROMAddress.

Referenced by BSP_EEPROM_WritePage().

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

Writes buffer of data to the I2C EEPROM.

Parameters:
- **pBuffer:** pointer to the buffer containing the data to be written to the EEPROM.
- **WriteAddr:** EEPROM's internal address to write to.
- **NumByteToWrite:** number of bytes to write to the EEPROM.

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

Definition at line 280 of file stm32f769i_discovery_eeprom.c.

References BSP_EEPROM_WritePage(), EEPROM_OK, and EEPROM_PAGESIZE.

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

Writes more than one byte to the EEPROM with a single WRITE cycle.

Note:
The number of bytes (combined to write start address) must not cross the EEPROM page boundary. This function can only write into the boundaries of an EEPROM page. This function doesn't check on boundaries condition (in this driver the function BSP_EEPROM_WriteBuffer() which calls
**BSP_EEPROM_WritePage()** is responsible of checking on Page boundaries).

**Parameters:**
- **pBuffer,:** pointer to the buffer containing the data to be written to the EEPROM.
- **WriteAddr,:** EEPROM’s internal address to write to.
- **NumByteToWrite,:** pointer to the variable holding number of bytes to be written into the EEPROM.

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

This function just configure the communication and enable the DMA channel to transfer data. Meanwhile, the user application may perform other tasks in parallel.

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

Definition at line 245 of file *stm32f769i_discovery_eeprom.c*.

References **BSP_EEPROM_TIMEOUT_UserCallback()**, **BSP_EEPROM_WaitEepromStandbyState()**, **EEPROM_FAIL**, **EEPROM_IO_WriteData()**, **EEPROM_OK**, **EEPROMAddress**, and **EEPROMDataWrite**.

Referenced by **BSP_EEPROM_WriteBuffer()**.

```c
void EEPROM_IO_Init ( void )
```

Initializes peripherals used by the I2C EEPROM driver.
HAL_StatusTypeDef EEPROM_IO_IsDeviceReady ( uint16_t DevAddress, 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**

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

Definition at line 659 of file stm32f769i_discovery.c.

References *hI2cExtHandler*, and I2Cx_IsDeviceReady().

Referenced by BSP_EEPROM_Init(), and BSP_EEPROM_WaitEepromStandbyState().
Read data from I2C EEPROM driver in using DMA channel.

**Parameters:**
- **DevAddress**: Target device address
- **MemAddress**: Internal memory address
- **pBuffer**: Pointer to data buffer
- **BufferSize**: Amount of data to be read

**Return values:**
- **HAL** status

Definition at line 647 of file `stm32f769i_discovery.c`.

References `hI2cExtHandler` and `I2Cx_ReadMultiple()`.

Referenced by `BSP_EEPROM_ReadBuffer()`.

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

Write data to I2C EEPROM driver in using DMA channel.

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

**Return values:**
- **HAL** status

Definition at line 634 of file `stm32f769i_discovery.c`. 
References `hI2cExtHandler`, and `I2Cx_WriteMultiple()`.

Referenced by `BSP_EEPROM_WritePage()`.
## STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td>Functions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### BSP Public Functions

**STM32F769I_DISCOVERY LOW LEVEL**
## Functions

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint32_t</td>
<td><strong>BSP_GetVersion</strong> (void)</td>
<td>This method returns the STM32F769I Discovery BSP Driver revision.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_LED_Init</strong> (Led_TypeDef Led)</td>
<td>Configures LED GPIO.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_LED_DeInit</strong> (Led_TypeDef Led)</td>
<td>DeInit LEDs.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_LED_On</strong> (Led_TypeDef Led)</td>
<td>Turns selected LED On.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_LED_Off</strong> (Led_TypeDef Led)</td>
<td>Turns selected LED Off.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_LED_Toggle</strong> (Led_TypeDef Led)</td>
<td>Toggles the selected LED.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_PB_Init</strong> (Button_TypeDef Button, ButtonMode_TypeDef Button_Mode)</td>
<td>Configures button GPIO and EXTI Line.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_PB_DeInit</strong> (Button_TypeDef Button)</td>
<td>Push Button DeInit.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><strong>BSP_PB_GetState</strong> (Button_TypeDef Button)</td>
<td>Returns the selected button state.</td>
</tr>
</tbody>
</table>
Function Documentation

`uint32_t BSP_GetVersion ( void )`

This method returns the STM32F769I Discovery BSP Driver revision.

**Return values:**

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

Definition at line 163 of file `stm32f769i_discovery.c`.

References `__STM32F769I_DISCOVERY_BSP_VERSION`.

`void BSP_LED_DeInit ( Led_TypeDef Led )`

DeInit LEDs.

**Parameters:**

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

- LED1
- LED2

**Note:**

Led DeInit does not disable the GPIO clock

**Return values:**

`None`

Definition at line 201 of file `stm32f769i_discovery.c`.

References `GPIO_PIN`, and `GPIO_PORT`.

`void BSP_LED_Init ( Led_TypeDef Led )`
Configures LED GPIO.

**Parameters:**

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

- LED1
- LED2

**Return values:**

None

Definition at line 176 of file `stm32f769i_discovery.c`.

References GPIO_PIN, GPIO_PORT, and LEDx_GPIO_CLK_ENABLE.

```c
void BSP_LED_Off (Led_TypeDef Led)
```

Turns selected LED Off.

**Parameters:**

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

- LED1
- LED2

**Return values:**

None

Definition at line 234 of file `stm32f769i_discovery.c`.

References GPIO_PIN, and GPIO_PORT.

```c
void BSP_LED_On (Led_TypeDef Led)
```
Turns selected LED On.

**Parameters:**

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

**Return values:**

- **None**

Definition at line 221 of file `stm32f769i_discovery.c`.

References `GPIO_PIN`, and `GPIO_PORT`.

```c
void BSP_LED_Toggle (Led_TypeDef Led)
```

Toggles the selected LED.

**Parameters:**

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

**Return values:**

- **None**

Definition at line 247 of file `stm32f769i_discovery.c`.

References `GPIO_PIN`, and `GPIO_PORT`.

```c
void BSP_PB_DeInit (Button_TypeDef Button)
```
Push Button DeInit.

**Parameters:**

- **Button,** Button to be configured This parameter can be one of the following values:
  - BUTTON_WAKEUP: Wakeup Push Button
  - BUTTON_USER: User Push Button

**Note:**

PB DeInit does not disable the GPIO clock

**Return values:**

- **None**

Definition at line 308 of file *stm32f769i_discovery.c*.

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

```c
uint32_t BSP_PB_GetState ( Button_TypeDef Button )
```

Returns the selected button state.

**Parameters:**

- **Button,** Button to be checked This parameter can be one of the following values:
  - BUTTON_WAKEUP: Wakeup Push Button
  - BUTTON_USER: User Push Button

**Return values:**

- **The** Button GPIO pin value

Definition at line 326 of file *stm32f769i_discovery.c*.

References **BUTTON_PIN**, and **BUTTON_PORT**.
void BSP_PB_Init ( Button_TypeDef Button, ButtonMode_TypeDef Button_Mode )

Configures button GPIO and EXTI Line.

Parameters:

- **Button,:** Button to be configured This parameter can be one of the following values:
  - BUTTON_WAKEUP: Wakeup Push Button
  - BUTTON_USER: User Push Button

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

Return values:

- None

Definition at line 265 of file stm32f769i_discovery.c.

References BUTTON_GPIO_CLK_ENABLE, BUTTON_IRQHandler, BUTTON_MODE_EXTI, BUTTON_MODE_GPIO, BUTTON_PIN, and BUTTON_PORT.
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td>Functions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

STM32F769I Discovery
Low Level Exported Functions

STM32F769I Discovery Low Level Exported Types
## Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint32_t</td>
<td><code>BSP_GetVersion</code> (void)</td>
<td>This method returns the STM32F769I Discovery BSP Driver revision.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_LED_Init</code> (Led_TypeDef Led)</td>
<td>Configures LED GPIO.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_LED_DeInit</code> (Led_TypeDef Led)</td>
<td>DeInit LEDs.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_LED_On</code> (Led_TypeDef Led)</td>
<td>Turns selected LED On.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_LED_Off</code> (Led_TypeDef Led)</td>
<td>Turns selected LED Off.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_LED_Toggle</code> (Led_TypeDef Led)</td>
<td>Toggles the selected LED.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_PB_Init</code> (Button_TypeDef Button, ButtonMode_TypeDef Button_Mode)</td>
<td>Configures button GPIO and EXTI Line.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_PB_DeInit</code> (Button_TypeDef Button)</td>
<td>Push Button DeInit.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><code>BSP_PB_GetState</code> (Button_TypeDef Button)</td>
<td>Returns the selected button state.</td>
</tr>
</tbody>
</table>
## Function Documentation

**uint32_t BSP_GetVersion ( void )**

This method returns the STM32F769I Discovery BSP Driver revision.

**Return values:**

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

Definition at line 163 of file `stm32f769i_discovery.c`.

References `__STM32F769I_DISCOVERY_BSP_VERSION`.

**void BSP_LED_DeInit ( Led_TypeDef Led )**

DeInit LEDs.

**Parameters:**

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

**Note:**

Led DeInit does not disable the GPIO clock

**Return values:**

- **None**

Definition at line 201 of file `stm32f769i_discovery.c`.

References `GPIO_PIN`, and `GPIO_PORT`.

**void BSP_LED_Init ( Led_TypeDef Led )**
Configures LED GPIO.

Parameters:

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

- LED1
- LED2

Return values:

None

Definition at line 176 of file stm32f769i_discovery.c.

References GPIO_PIN, GPIO_PORT, and LEDx_GPIO_CLK_ENABLE.

```c
void BSP_LED_Off ( Led_TypeDef Led )
```

Turns selected LED Off.

Parameters:

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

- LED1
- LED2

Return values:

None

Definition at line 234 of file stm32f769i_discovery.c.

References GPIO_PIN, and GPIO_PORT.

```c
void BSP_LED_On ( Led_TypeDef Led )
```
Turns selected LED On.

**Parameters:**

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

**Return values:**

- **None**

Definition at line 221 of file *stm32f769i_discovery.c*.

References `GPIO_PIN`, and `GPIO_PORT`.

```c
void BSP_LED_Toggle ( Led_TypeDef  Led )
```

Toggles the selected LED.

**Parameters:**

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

**Return values:**

- **None**

Definition at line 247 of file *stm32f769i_discovery.c*.

References `GPIO_PIN`, and `GPIO_PORT`.

```c
void BSP_PB_DeInit ( Button_TypeDef  Button )
```
Push Button DeInit.

Parameters:

- **Button**, Button to be configured This parameter can be one of the following values:
  - BUTTON_WAKEUP: Wakeup Push Button
  - BUTTON_USER: User Push Button

Note: PB DeInit does not disable the GPIO clock

Return values:

None

Definition at line 308 of file stm32f769i_discovery.c.

References BUTTON_IRQn, BUTTON_PIN, and BUTTON_PORT.

```c
uint32_t BSP_PB_GetState ( Button_TypeDef Button )
```

Returns the selected button state.

Parameters:

- **Button**, Button to be checked This parameter can be one of the following values:
  - BUTTON_WAKEUP: Wakeup Push Button
  - BUTTON_USER: User Push Button

Return values:

The Button GPIO pin value

Definition at line 326 of file stm32f769i_discovery.c.

References BUTTON_PIN, and BUTTON_PORT.
```c
void BSP_PB_Init ( Button_TypeDef Button,
                 ButtonMode_TypeDef Button_Mode )
```

Configures button GPIO and EXTI Line.

**Parameters:**
- **Button,:** Button to be configured. This parameter can be one of the following values:
  - BUTTON_WAKEUP: Wakeup Push Button
  - BUTTON_USER: User Push Button
- **Button_Mode,:** Button mode. This parameter can be one of the following values:
  - BUTTON_MODE_GPIO: Button will be used as simple IO
  - BUTTON_MODE_EXTI: Button will be connected to EXTI line with interrupt generation capability

**Return values:**
- None

Definition at line 265 of file `stm32f769i_discovery.c`.

References BUTTON_GPIO_CLK_ENABLE, BUTTON_IRQn, BUTTON_MODE_EXTI, BUTTON_MODE_GPIO, BUTTON_PIN, and BUTTON_PORT.
# LCD Exported Functions

| STM32F769I_DISCOVERY_LCD | STM32F769I_DISCOVERY_LCD_Exported_MACRO |
## Functions

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td><code>BSP_LCD_Init</code> (void)</td>
<td>Initializes the DSI LCD.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><code>BSP_LCD_InitEx</code> (LCD_OrientationTypeDef orientation)</td>
<td>Initializes the DSI LCD.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_LCD_Reset</code> (void)</td>
<td>BSP LCD Reset Hw reset the LCD DSI activating its XRES signal (active low for some time) and desactivating it later.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><code>BSP_LCD_GetXSize</code> (void)</td>
<td>Gets the LCD X size.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><code>BSP_LCD_GetYSize</code> (void)</td>
<td>Gets the LCD Y size.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_LCD_SetXSize</code> (uint32_t imageWidthPixels)</td>
<td>Set the LCD X size.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_LCD_SetYSize</code> (uint32_t imageHeightPixels)</td>
<td>Set the LCD Y size.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_LCD_LayerDefaultInit</code> (uint16_t LayerIndex, uint32_t FB_Address)</td>
<td>Initializes the LCD layers.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_LCD_SelectLayer</code> (uint32_t LayerIndex)</td>
<td>Selects the LCD Layer.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_LCD_SetLayerVisible</code> (uint32_t LayerIndex, FunctionalState State)</td>
<td>Sets an LCD Layer visible.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_LCD_SetTransparency</code> (uint32_t LayerIndex, uint8_t Transparency)</td>
<td>Configures the transparency.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_LCD_SetLayerAddress</code> (uint32_t LayerIndex, uint32_t Address)</td>
<td>Sets an LCD layer frame buffer address.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>void <strong>BSP_LCD_SetLayerWindow</strong> (uint16_t LayerIndex, uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height)</td>
<td>Sets display window.</td>
<td></td>
</tr>
<tr>
<td>void <strong>BSP_LCD_SetColorKeying</strong> (uint32_t LayerIndex, uint32_t RGBValue)</td>
<td>Configures and sets the color keying.</td>
<td></td>
</tr>
<tr>
<td>void <strong>BSP_LCD_ResetColorKeying</strong> (uint32_t LayerIndex)</td>
<td>Disables the color keying.</td>
<td></td>
</tr>
<tr>
<td>void <strong>BSP_LCD_SetTextColor</strong> (uint32_t Color)</td>
<td>Sets the LCD text color.</td>
<td></td>
</tr>
<tr>
<td>uint32_t <strong>BSP_LCD_GetTextColor</strong> (void)</td>
<td>Gets the LCD text color.</td>
<td></td>
</tr>
<tr>
<td>void <strong>BSP_LCD_SetBackColor</strong> (uint32_t Color)</td>
<td>Sets the LCD background color.</td>
<td></td>
</tr>
<tr>
<td>uint32_t <strong>BSP_LCD_GetBackColor</strong> (void)</td>
<td>Gets the LCD background color.</td>
<td></td>
</tr>
<tr>
<td>void <strong>BSP_LCD_SetFont</strong> (sFONT *fonts)</td>
<td>Sets the LCD text font.</td>
<td></td>
</tr>
<tr>
<td>sFONT * <strong>BSP_LCD_GetFont</strong> (void)</td>
<td>Gets the LCD text font.</td>
<td></td>
</tr>
<tr>
<td>uint32_t <strong>BSP_LCD_ReadPixel</strong> (uint16_t Xpos, uint16_t Ypos)</td>
<td>Reads an LCD pixel.</td>
<td></td>
</tr>
<tr>
<td>void <strong>BSP_LCD_Clear</strong> (uint32_t Color)</td>
<td>Clears the whole currently active layer of LTDC.</td>
<td></td>
</tr>
<tr>
<td>void <strong>BSP_LCD_ClearStringLine</strong> (uint32_t Line)</td>
<td>Clears the selected line in currently active layer.</td>
<td></td>
</tr>
<tr>
<td>void <strong>BSP_LCD_DisplayChar</strong> (uint16_t Xpos, uint16_t Ypos, uint8_t Ascii)</td>
<td>Displays one character in currently active layer.</td>
<td></td>
</tr>
<tr>
<td>void <strong>BSP_LCD_DisplayStringAt</strong> (uint16_t Xpos, uint16_t Ypos, uint8_t *Text, Text_AlignModeTypdef Mode)</td>
<td>Displays characters in currently active layer.</td>
<td></td>
</tr>
</tbody>
</table>
| void **BSP_LCD_DisplayStringAtLine** (uint16_t Line,
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void uint8_t *ptr)</td>
<td>Displays a maximum of 60 characters on the LCD.</td>
</tr>
<tr>
<td>void BSP_LCD_DrawHLine</td>
<td>Draws an horizontal line in currently active layer.</td>
</tr>
<tr>
<td>void BSP_LCD_DrawVLine</td>
<td>Draws a vertical line in currently active layer.</td>
</tr>
<tr>
<td>void BSP_LCD_DrawLine</td>
<td>Draws an uni-line (between two points) in currently active layer.</td>
</tr>
<tr>
<td>void BSP_LCD_DrawRect</td>
<td>Draws a rectangle in currently active layer.</td>
</tr>
<tr>
<td>void BSP_LCD_DrawCircle</td>
<td>Draws a circle in currently active layer.</td>
</tr>
<tr>
<td>void BSP_LCD_DrawPolygon</td>
<td>Draws an poly-line (between many points) in currently active layer.</td>
</tr>
<tr>
<td>void BSP_LCD_DrawEllipse</td>
<td>Draws an ellipse on LCD in currently active layer.</td>
</tr>
<tr>
<td>void BSP_LCD_DrawBitmap</td>
<td>Draws a bitmap picture loaded in the internal Flash (32 bpp) in currently active layer.</td>
</tr>
<tr>
<td>void BSP_LCD_FillRect</td>
<td>Draws a full rectangle in currently active layer.</td>
</tr>
<tr>
<td>void BSP_LCD_FillCircle</td>
<td>Draws a circle in currently active layer.</td>
</tr>
<tr>
<td>Function Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>BSP_LCD_FillPolygon</code> (pPoint Points, uint16_t PointCount)</td>
<td>Draws a full poly-line (between many points) in currently active layer.</td>
</tr>
<tr>
<td><code>BSP_LCD_FillEllipse</code> (int Xpos, int Ypos, int XRadius, int YRadius)</td>
<td>Draws a full ellipse in currently active layer.</td>
</tr>
<tr>
<td><code>BSP_LCD_DisplayOn</code> (void)</td>
<td>Switch back on the display if was switched off by previous call of <code>BSP_LCD_DisplayOff()</code>.</td>
</tr>
<tr>
<td><code>BSP_LCD_DisplayOff</code> (void)</td>
<td>Switch Off the display.</td>
</tr>
<tr>
<td><code>BSP_LCD_SetBrightness</code> (uint8_t BrightnessValue)</td>
<td>Set the brightness value.</td>
</tr>
<tr>
<td><code>DSI_IO_WriteCmd</code> (uint32_t NbrParams, uint8_t *pParams)</td>
<td>DCS or Generic short/long write command.</td>
</tr>
<tr>
<td><code>BSP_LCD_MspDeInit</code> (void)</td>
<td>De-Initializes the BSP LCD Msp Application can surcharge if needed this function implementation.</td>
</tr>
<tr>
<td><code>BSP_LCD_MspInit</code> (void)</td>
<td>Initialize the BSP LCD Msp.</td>
</tr>
<tr>
<td><code>BSP_LCD_DrawPixel</code> (uint16_t Xpos, uint16_t Ypos, uint32_t RGB_Code)</td>
<td>Draws a pixel on LCD.</td>
</tr>
<tr>
<td><code>BSP_LCD_HDMIInitEx</code> (uint8_t format)</td>
<td>Draws a character on LCD.</td>
</tr>
</tbody>
</table>

`BSP_LCD_HDMIInitEx` returns the ID of connected screen by checking the HDMI (adv7533 component) ID or LCD DSI (via TS ID) ID.
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static void FillTriangle(uint16_t x1, uint16_t x2, uint16_t x3, uint16_t y1, uint16_t y2, uint16_t y3)</code></td>
<td>Fills a triangle (between 3 points).</td>
</tr>
<tr>
<td><code>static void LL_FillBuffer(uint32_t LayerIndex, void *pDest, uint32_t xSize, uint32_t ySize, uint32_t OffLine, uint32_t ColorIndex)</code></td>
<td>Fills a buffer.</td>
</tr>
<tr>
<td><code>static void LL_ConvertLineToARGB8888(void *pSrc, void *pDest, uint32_t xSize, uint32_t ColorMode)</code></td>
<td>Converts a line to an ARGB8888 pixel format.</td>
</tr>
</tbody>
</table>
Function Documentation

`void BSP_LCD_Clear ( uint32_t Color )`

Clears the whole currently active layer of LTDC.

**Parameters:**

  - **Color,** Color of the background

Definition at line 956 of file `stm32f769i_discovery_lcd.c`.

References `ActiveLayer`, `BSP_LCD_GetXSize()`, `BSP_LCD_GetYSize()`, `hltdc_discovery`, and `LL_FillBuffer()`.

`void BSP_LCD_ClearStringLine ( uint32_t Line )`

Clears the selected line in currently active layer.

**Parameters:**

  - **Line,** Line to be cleared

Definition at line 966 of file `stm32f769i_discovery_lcd.c`.

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

`void BSP_LCD_DisplayChar ( uint16_t Xpos, uint16_t Ypos, uint8_t Ascii )`
Displays one character in currently active layer.

**Parameters:**
- **Xpos,** Start column address
- **Ypos,** Line where to display the character shape.
- **Ascii,** Character ascii code This parameter must be a number between Min_Data = 0x20 and Max_Data = 0x7E

Definition at line 985 of file *stm32f769i_discovery_lcd.c*.

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

Referenced by `BSP_LCD_DisplayStringAt()`.

```c
void BSP_LCD_DisplayOff ( void )
```

Switch Off the display.

Enter DSI ULPM mode if was allowed and configured in Dsi Configuration.

Definition at line 1553 of file *stm32f769i_discovery_lcd.c*.

References `hdsi_discovery`, and `hdsivideo_handle`.

```c
void BSP_LCD_DisplayOn ( void )
```

Switch back on the display if was switched off by previous call of `BSP_LCD_DisplayOff()`.

Exit DSI ULPM mode if was allowed and configured in Dsi Configuration.
Definition at line 1530 of file stm32f769i_discovery_lcd.c.

References hdsi_discovery, and hdsivideo_handle.

```c
void BSP_LCD_DisplayStringAt ( uint16_t Xpos,
                               uint16_t Ypos,
                               uint8_t * Text,
                               Text_AlignModeTypeDef Mode
                           )

Displays characters in currently active layer.

Parameters:

  Xpos,: X position (in pixel)
  Ypos,: Y position (in pixel)
  Text,: Pointer to string to display on LCD
  Mode,: Display mode This parameter can be one of the following values:
         • CENTER_MODE
         • RIGHT_MODE
         • LEFT_MODE

Definition at line 1002 of file stm32f769i_discovery_lcd.c.

References ActiveLayer, BSP_LCD_DisplayChar(),
BSP_LCD_GetXSize(), CENTER_MODE, LEFT_MODE,
LCD_DrawPropTypeDef::pFont, and RIGHT_MODE.

Referenced by BSP_LCD_DisplayStringAtLine().

```c
void BSP_LCD_DisplayStringAtLine ( uint16_t Line,
                                   uint8_t * ptr
                               )
```
Displays a maximum of 60 characters on the LCD.

**Parameters:**
- **Line:** Line where to display the character shape
- **ptr:** Pointer to string to display on LCD

Definition at line 1064 of file `stm32f769i_discovery_lcd.c`.

References **BSP_LCD_DisplayStringAt()**, and **LEFT_MODE**.

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

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

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

Definition at line 1306 of file `stm32f769i_discovery_lcd.c`.

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

```c
void BSP_LCD_DrawCircle ( uint16_t Xpos,
                         uint16_t Ypos,
                         uint16_t Radius
)```
Draws a circle in currently active layer.

**Parameters:**
- **Xpos:** X position
- **Ypos:** Y position
- **Radius:** Circle radius

Definition at line 1201 of file *stm32f769i_discovery_lcd.c*.

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

Referenced by **BSP_LCD_FillCircle()**.

```c
void BSP_LCD_DrawEllipse ( int Xpos,
                          int Ypos,
                          int XRadius,
                          int YRadius
                    )
```

Draws an ellipse on LCD in currently active layer.

**Parameters:**
- **Xpos:** X position
- **Ypos:** Y position
- **XRadius:** Ellipse X radius
- **YRadius:** Ellipse Y radius

Definition at line 1274 of file *stm32f769i_discovery_lcd.c*.

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

```c
void BSP_LCD_DrawHLine ( uint16_t Xpos,
                         uint16_t Ypos,
```
uint16_t  Length
)

Draws an horizontal line in currently active layer.

Parameters:
   Xpos, : X position
   Ypos, : Y position
   Length, : Line length

Definition at line 1075 of file stm32f769i_discovery_lcd.c.

References ActiveLayer, BSP_LCD_GetXSize(), hltdc_discovery, and LL_FillBuffer().

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

void BSP_LCD_DrawLine ( uint16_t  x1,
    uint16_t  y1,
    uint16_t  x2,
    uint16_t  y2
)

Draws an uni-line (between two points) in currently active layer.

Parameters:
   x1, : Point 1 X position
   y1, : Point 1 Y position
   x2, : Point 2 X position
   y2, : Point 2 Y position

Definition at line 1110 of file stm32f769i_discovery_lcd.c.

References ABS, ActiveLayer, and BSP_LCD_DrawPixel().
Referenced by `BSP_LCD_DrawPolygon()`, and `FillTriangle()`.

```c
void BSP_LCD_DrawPixel ( uint16_t Xpos,
                        uint16_t Ypos,
                        uint32_t RGB_Code
)  
```

Draws a pixel on LCD.

**Parameters:**
- **Xpos:** X position
- **Ypos:** Y position
- **RGB_Code:** Pixel color in ARGB mode (8-8-8-8)

Definition at line **1717** of file `stm32f769i_discovery_lcd.c`.

References `ActiveLayer`, `BSP_LCD_GetXSize()`, and `hltdc_discovery`.

Referenced by `BSP_LCD_DrawCircle()`, `BSP_LCD_DrawEllipse()`, `BSP_LCD_DrawLine()`, and `DrawChar()`.

```c
void BSP_LCD_DrawPolygon ( pPoint Points,
                           uint16_t PointCount
)  
```

Draws an poly-line (between many points) in currently active layer.

**Parameters:**
- **Points:** Pointer to the points array
- **PointCount:** Number of points

Definition at line **1247** of file `stm32f769i_discovery_lcd.c`. 
References **BSP_LCD_DrawLine()**, **Point::X**, and **Point::Y**.

```c
void BSP_LCD_DrawRect ( uint16_t Xpos,
                        uint16_t Ypos,
                        uint16_t Width,
                        uint16_t Height )
```

Draws a rectangle in currently active layer.

**Parameters:**
- **Xpos:** X position
- **Ypos:** Y position
- **Width:** Rectangle width
- **Height:** Rectangle height

Definition at line 1184 of file *stm32f769i_discovery_lcd.c*.

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

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

Draws a vertical line in currently active layer.

**Parameters:**
- **Xpos:** X position
- **Ypos:** Y position
- **Length:** Line length

Definition at line 1092 of file *stm32f769i_discovery_lcd.c*. 
References ActiveLayer, BSP_LCD_GetXSize(), hltdc_discovery, and LL_FillBuffer().

Referenced by BSP_LCD_DrawRect().

```c
void BSP_LCD_FillCircle ( uint16_t Xpos,
                          uint16_t Ypos,
                          uint16_t Radius
                     )
```

Draws a full circle in currently active layer.

**Parameters:**
- **Xpos,** X position
- **Ypos,** Y position
- **Radius,** Circle radius

Definition at line 1386 of file stm32f769i_discovery_lcd.c.

References ActiveLayer, BSP_LCD_DrawCircle(), BSP_LCD_DrawHLine(), and BSP_LCD_SetTextColor().

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

Draws a full ellipse in currently active layer.

**Parameters:**
- **Xpos,** X position
- **Ypos,** Y position
XRadius, : Ellipse X radius
YRadius, : Ellipse Y radius

Definition at line 1500 of file stm32f769i_discovery_lcd.c.

References BSP_LCD_DrawHLine().

```c
void BSP_LCD_FillPolygon ( pPoint Points,
                           uint16_t PointCount
                      )
```

Draws a full poly-line (between many points) in currently active layer.

**Parameters:**
- Points, : Pointer to the points array
- PointCount, : Number of points

Definition at line 1433 of file stm32f769i_discovery_lcd.c.

References FillTriangle(), POLY_X, POLY_Y, Point::X, and Point::Y.

```c
void BSP_LCD_FillRect ( uint16_t Xpos,
                        uint16_t Ypos,
                        uint16_t Width,
                        uint16_t Height
                   )
```

Draws a full rectangle in currently active layer.

**Parameters:**
- Xpos, : X position
- Ypos, : Y position
- Width, : Rectangle width
**Height:** Rectangle height

Definition at line 1366 of file stm32f769i_discovery_lcd.c.

References ActiveLayer, BSP_LCD_GetXSize(), BSP_LCD_SetTextColor(), hltdc_discovery, and LL_FillBuffer().

Referenced by BSP_LCD_ClearStringLine().

```c
uint32_t BSP_LCD_GetBackColor ( void )
```

Gets the LCD background color.

**Return values:**
- **Used** background color

Definition at line 893 of file stm32f769i_discovery_lcd.c.

References ActiveLayer, and LCD_DrawPropTypeDef::BackColor.

```c
sFONT * BSP_LCD_GetFont ( void )
```

Gets the LCD text font.

**Return values:**
- **Used** layer font

Definition at line 911 of file stm32f769i_discovery_lcd.c.

References ActiveLayer, and LCD_DrawPropTypeDef::pFont.

```c
uint32_t BSP_LCD_GetTextColor ( void )
```

Gets the LCD text color.
Return values:  
**Used** text color.

Definition at line 875 of file `stm32f769i_discovery_lcd.c`.

References `ActiveLayer`, and `LCD_DrawPropTypeDef::TextColor`.

```c
uint32_t BSP_LCD_GetXSize ( void )
```

Gets the LCD X size.

**Return values:**  
**Used** LCD X size

Definition at line 697 of file `stm32f769i_discovery_lcd.c`.

References `lcd_x_size`.


```c
uint32_t BSP_LCD_GetYSize ( void )
```

Gets the LCD Y size.

**Return values:**  
**Used** LCD Y size

Definition at line 706 of file `stm32f769i_discovery_lcd.c`.

References `lcd_y_size`.

Referenced by `BSP_LCD_Clear()`, and
BSP_LCD_LayerDefaultInit().

uint8_t BSP_LCD_HDMIInitEx ( uint8_t format )

Referenced by BSP_LCD_InitEx().

uint8_t BSP_LCD_Init ( void )

Initializes the DSI LCD.

Return values:
   LCD state

Definition at line 257 of file stm32f769i_discovery_lcd.c.

References BSP_LCD_InitEx(), and LCD_ORIENTATION_LANDSCAPE.

uint8_t BSP_LCD_InitEx ( LCD_OrientationTypeDef orientation )

Initializes the DSI LCD.

The initialization is done as below:

- DSI PLL initialization
- DSI initialization
- LTDC initialization
- OTM8009A LCD Display IC Driver initialization

Parameters:
   orientation: LCD orientation, can be
                 LCD_ORIENTATION_PORTRAIT or
                 LCD_ORIENTATION_LANDSCAPE

Return values:
   LCD state
< LcdClk = 27429 kHz
< Vertical start active time in units of lines
< Vertical Back Porch time in units of lines
< Vertical Front Porch time in units of lines
< Vertical Active time in units of lines = imageSize Y in pixels to display
< Horizontal start active time in units of LcdClk
< Horizontal Back Porch time in units of LcdClk
< Horizontal Front Porch time in units of LcdClk
< Horizontal Active time in units of LcdClk = imageSize X in pixels to display

LCD clock configuration Note: The following values should not be changed as the PLLSAI is also used to clock the USB FS
PLLSAI_VCO Input = HSE_VALUE/PLL_M = 1 Mhz PLLSAI_VCO
Output = PLLSAI_VCO Input * PLLSAIN = 384 Mhz PLLLCDCLK = PLLSAI_VCO Output/PLLSAIR = 384 MHz / 7 = 54.85 MHz LTDC
clock frequency = PLLLCDCLK / LTDC_PLLSAI_DIVR_2 = 54.85 MHz / 2 = 27.429 MHz

Definition at line 272 of file stm32f769i_discovery_lcd.c.

References BSP_LCD_HDMIIInitEx(), BSP_LCD_MspInit(), BSP_LCD_Reset(), BSP_LCD_SetFont(), BSP_SDRAM_Init(),
HDMI_FORMAT_720_576, hdsi_discovery, hdsivideo_handle,
hltdc_discovery, LCD_DEFAULT_FONT, LCD_DSI_ID,
LCD_DSI_PIXEL_DATA_FMT_RGB888, LCD_ERROR,
LCD_IO_GetID(), LCD_OK, LCD_ORIENTATION_PORTRAIT,
LCD_OTM8009A_ID, lcd_x_size, and lcd_y_size.

Referenced by BSP_LCD_Init().
**void** **BSP_LCD_LayerDefaultInit** (uint16_t **LayerIndex**, uint32_t **FB_Address**)

 Initializes the LCD layers.

 **Parameters:**

- **LayerIndex:** Layer foreground or background
- **FB_Address:** Layer frame buffer

 **Return values:**

None

Definition at line 737 of file *stm32f769i_discovery_lcd.c*.

References **LCD_DrawPropTypeDef::BackColor**, **BSP_LCD_GetXSize()**, **BSP_LCD_GetYSize()**, **hltdc_discovery**, **LCD_COLOR_BLACK**, **LCD_COLOR_WHITE**, **LCD_LayerCfgTypeDef**, **LCD_DrawPropTypeDef::pFont**, and **LCD_DrawPropTypeDef::TextColor**.

**void** **BSP_LCD_MspDeInit** ( void )

De-Initializes the BSP LCD Msp Application can surcharge if needed this function implementation.

Disable IRQ of LTDC IP

Disable IRQ of DMA2D IP

Disable IRQ of DSI IP

Force and let in reset state LTDC, DMA2D and DSI Host + Wrapper IPs
Disable the LTDC, DMA2D and DSI Host and Wrapper clocks
Definition at line 1649 of file stm32f769i_discovery_lcd.c.

```c
void BSP_LCD_MspInit ( void )
```

Initialize the BSP LCD Msp.
Application can surcharge if needed this function implementation
Enable the LTDC clock
Toggle Sw reset of LTDC IP
Enable the DMA2D clock
Toggle Sw reset of DMA2D IP
Enable DSI Host and wrapper clocks
Soft Reset the DSI Host and wrapper
NVIC configuration for LTDC interrupt that is now enabled
NVIC configuration for DMA2D interrupt that is now enabled
NVIC configuration for DSI interrupt that is now enabled
Definition at line 1675 of file stm32f769i_discovery_lcd.c.
Referenced by BSP_LCD_InitEx().

```c
uint32_t BSP_LCD_ReadPixel ( uint16_t Xpos,
                              uint16_t Ypos
                      )
```

Reads an LCD pixel.
Parameters:
  Xpos,:  X position
  Ypos,:  Y position

Return values:
  RGB  pixel color

Definition at line 922 of file stm32f769i_discovery_lcd.c.

References ActiveLayer, BSP_LCD_GetXSize(), and hltdc_discovery.

```c
void BSP_LCD_Reset ( void )
```

BSP LCD Reset Hw reset the LCD DSI activating its XRES signal (active low for some time) and desactivating it later.

Definition at line 667 of file stm32f769i_discovery_lcd.c.

Referenced by BSP_LCD_InitEx().

```c
void BSP_LCD_ResetColorKeying ( uint32_t LayerIndex )
```

Disables the color keying.

Parameters:
  LayerIndex,:  Layer foreground or background

Definition at line 856 of file stm32f769i_discovery_lcd.c.

References hltdc_discovery.

```c
void BSP_LCD_SelectLayer ( uint32_t LayerIndex )
```
Selects the LCD Layer.

**Parameters:**
- **LayerIndex:** Layer foreground or background

Definition at line 770 of file *stm32f769i_discovery_lcd.c*.

References **ActiveLayer**.

```c
void BSP_LCD_SetBackColor ( uint32_t Color )
```

Sets the LCD background color.

**Parameters:**
- **Color:** Layer background color code ARGB(8-8-8-8)

Definition at line 884 of file *stm32f769i_discovery_lcd.c*.

References **ActiveLayer**, and **LCD_DrawPropTypeDef::BackColor**.

```c
void BSP_LCD_SetBrightness ( uint8_t BrightnessValue )
```

Set the brightness value.

**Parameters:**
- **BrightnessValue:** [00: Min (black), 100 Max]

Definition at line 1576 of file *stm32f769i_discovery_lcd.c*.

References **hdsi_discovery**, and **LCD_OTM8009A_ID**.

```c
void BSP_LCD_SetColorKeying ( uint32_t LayerIndex, uint32_t RGBValue
```
Configures and sets the color keying.

**Parameters:**
- **LayerIndex:** Layer foreground or background
- **RGBValue:** Color reference

Definition at line 845 of file `stm32f769i_discovery_lcd.c`.

References `hltdc_discovery`.

```c
void BSP_LCD_SetFont ( sFONT * fonts )
```

Sets the LCD text font.

**Parameters:**
- **fonts:** Layer font to be used

Definition at line 902 of file `stm32f769i_discovery_lcd.c`.

References `ActiveLayer`, and `LCD_DrawPropTypeDef::pFont`.

Referenced by `BSP_LCD_InitEx()`.

```c
void BSP_LCD_SetLayerAddress ( uint32_t LayerIndex, uint32_t Address )
```

Sets an LCD layer frame buffer address.

**Parameters:**
- **LayerIndex:** Layer foreground or background
- **Address:** New LCD frame buffer value
void BSP_LCD_SetLayerVisible ( uint32_t LayerIndex, FunctionalState State )

Sets an LCD Layer visible.

**Parameters:**
- **LayerIndex:** Visible Layer
- **State:** New state of the specified layer This parameter can be one of the following values:
  - ENABLE
  - DISABLE

void BSP_LCD_SetLayerWindow ( uint16_t LayerIndex, uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height )

Sets display window.

**Parameters:**
- **LayerIndex:** Layer index
- **Xpos:** LCD X position
- **Ypos:** LCD Y position
**Width,** LCD window width

**Height,** LCD window height

Definition at line 830 of file `stm32f769i_discovery_lcd.c`.

References `hltdc_discovery`.

```c
void BSP_LCD_SetTextColor ( uint32_t Color )
```

Sets the LCD text color.

**Parameters:**

- **Color,** Text color code ARGB(8-8-8-8)

Definition at line 866 of file `stm32f769i_discovery_lcd.c`.

References `ActiveLayer`, and `LCD_DrawPropTypeDef::TextColor`.

Referenced by `BSP_LCD_ClearStringLine()`, `BSP_LCD_FillCircle()` , and `BSP_LCD_FillRect()`.

```c
void BSP_LCD_SetTransparency ( uint32_t LayerIndex,
                               uint8_t Transparency
                           )
```

Configures the transparency.

**Parameters:**

- **LayerIndex,** Layer foreground or background.
- **Transparency,** Transparency This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF

Definition at line 803 of file `stm32f769i_discovery_lcd.c`. 
References **hltdc_discovery.**

```c
void BSP_LCD_SetXSize( uint32_t imageWidthPixels )
```

Set the LCD X size.

**Parameters:**
- `imageWidthPixels` : uint32_t image width in pixels unit

**Return values:**
- None

Definition at line 716 of file `stm32f769i_discovery_lcd.c`.

References **ActiveLayer**, and **hltdc_discovery**.

```c
void BSP_LCD_SetYSize( uint32_t imageHeightPixels )
```

Set the LCD Y size.

**Parameters:**
- `imageHeightPixels` : uint32_t image height in lines unit

Definition at line 725 of file `stm32f769i_discovery_lcd.c`.

References **ActiveLayer**, and **hltdc_discovery**.

```c
static void DrawChar( uint16_t Xpos,
                     uint16_t Ypos,
                     const uint8_t* c )
```

Draws a character on LCD.
Parameters:

- **Xpos:** Line where to display the character shape
- **Ypos:** Start column address
- **c:** Pointer to the character data

Definition at line 1730 of file `stm32f769i_discovery_lcd.c`.

References `ActiveLayer`, `BSP_LCD_DrawPixel()`, and `LCD_DrawPropTypeDef::pFont`.

Referenced by `BSP_LCD_DisplayChar()`.

```c
void DSI_IO_WriteCmd ( uint32_t NbrParams,
                       uint8_t * pParams
                     )
```

DCS or Generic short/long write command.

Parameters:

- **NbrParams:** Number of parameters. It indicates the write command mode: If inferior to 2, a long write command is performed else short.
- **pParams:** Pointer to parameter values table.

Return values:

- **HAL** status

Definition at line 1601 of file `stm32f769i_discovery_lcd.c`.

References `hdsi_discovery`, and `LCD_OTM8009A_ID`.

```c
static void FillTriangle ( uint16_t x1,
                          uint16_t x2,
                          uint16_t x3,
```
Fills a triangle (between 3 points).

**Parameters:**
- \( x_1, \) **Point** 1 X position
- \( y_1, \) **Point** 1 Y position
- \( x_2, \) **Point** 2 X position
- \( y_2, \) **Point** 2 Y position
- \( x_3, \) **Point** 3 X position
- \( y_3, \) **Point** 3 Y position

Definition at line 1788 of file *stm32f769i_discovery_lcd.c*.

References **ABS**, and **BSP_LCD_DrawLine**().

Referenced by **BSP_LCD_FillPolygon**().

```c
static uint16_t LCD_IO_GetID ( void ) [static]
```

Returns the ID of connected screen by checking the HDMI (adv7533 component) ID or LCD DSI (via TS ID) ID.

**Return values:**
- **LCD** ID

Definition at line 1618 of file *stm32f769i_discovery_lcd.c*.

References **HDMI_IO_Delay()**, **HDMI_IO_Init()**, **HDMI_IO_Read()**, **LCD_DSI_ADDRESS**, **LCD_DSI_ID**, and **LCD_DSI_ID_REG**.

Referenced by **BSP_LCD_InitEx**().
static void LL_ConvertLineToARGB8888 ( void * pSrc, void * pDst, uint32_t xSize, uint32_t ColorMode )

Converts a line to an ARGB8888 pixel format.

Parameters:
- **pSrc:** Pointer to source buffer
- **pDst:** Output color
- **xSize:** Buffer width
- **ColorMode:** Input color mode

Definition at line 1895 of file stm32f769i_discovery_lcd.c.

References hdma2d_discovery.

Referenced by BSP_LCD_DrawBitmap().

static void LL_FillBuffer ( uint32_t LayerIndex, void * pDst, uint32_t xSize, uint32_t ySize, uint32_t OffLine, uint32_t ColorIndex )

Fills a buffer.

Parameters:
- **LayerIndex:** Layer index
- **pDst:** Pointer to destination buffer
xSize,: Buffer width
ySize,: Buffer height
OffLine,: Offset
ColorIndex,: Color index

Definition at line 1865 of file stm32f769i_discovery_lcd.c.

References hdma2d_discovery.

Referenced by BSP_LCD_Clear(), BSP_LCD_DrawHLine(),
BSP_LCD_DrawVLine(), and BSP_LCD_FillRect().
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Directories</th>
<th>Defines</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F769I DISCOVERY LCD Exported Constants</td>
<td></td>
</tr>
</tbody>
</table>

STM32F769I_DISCOVERY_LCD
Defines

#define BSP_LCD_DMA2D_IRQHandler DMA2D_IRQHandler
#define BSP_LCD_DSI_IRQHandler DSI_IRQHandler
#define BSP_LCD_LTDC_IRQHandler LTDC_IRQHandler
#define BSP_LCD_LTDC_ER_IRQHandler LTDC_ER_IRQHandler
#define LCD_LayerCfgTypeDef LTDC_LayerCfgTypeDef
#define LCD_FB_START_ADDRESS ((uint32_t)0xC0000000)
  LCD_FB_StartAddress.
#define LTDC_MAX_LAYER_NUMBER ((uint32_t)2)
  Maximum number of LTDC layers.
#define LTDC_ACTIVE_LAYER_BACKGROUND ((uint32_t)0)
  LTDC Background layer index.
#define LTDC_ACTIVE_LAYER_FOREGROUND ((uint32_t)1)
  LTDC Foreground layer index.
#define LTDC_NB_OF_LAYERS ((uint32_t)2)
  Number of LTDC layers.
#define LTDC_DEFAULT_ACTIVE_LAYER LTDC_ACTIVE_LAYER_FOREGROUND
  LTDC Default used layer index.
#define LCD_OK 0x00
  LCD status structure definition.
#define LCD_ERROR 0x01
#define LCD_TIMEOUT 0x02
#define LCD_OTM8009A_ID ((uint32_t)0)
  LCD Display OTM8009A DSI Virtual Channel ID.
#define HDMI_ADV7533_ID ((uint32_t)0)
  HDMI ADV7533 DSI Virtual Channel ID.
#define HDMI_FORMAT_720_480 ((uint8_t)0x00)
  /*720_480 format choice of HDMI display */
  HDMI Format.
#define HDMI_FORMAT_720_576 ((uint8_t)0x01)
  /*720_576 format choice of HDMI display*/
#define LCD_COLOR_BLUE ((uint32_t)0xFF0000FF)
LCD color definitions values in ARGB8888 format.

<table>
<thead>
<tr>
<th>Definition</th>
<th>Value</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#define LCD_COLOR_GREEN</code></td>
<td>((uint32_t) 0xFF00FF00)</td>
<td>Green value in ARGB8888 format.</td>
</tr>
<tr>
<td><code>#define LCD_COLOR_RED</code></td>
<td>((uint32_t) 0xFFFF0000)</td>
<td>Red value in ARGB8888 format.</td>
</tr>
<tr>
<td><code>#define LCD_COLOR_CYAN</code></td>
<td>((uint32_t) 0xFF00FFFF)</td>
<td>Cyan value in ARGB8888 format.</td>
</tr>
<tr>
<td><code>#define LCD_COLOR_MAGENTA</code></td>
<td>((uint32_t) 0xFFFF00FF)</td>
<td>Magenta value in ARGB8888 format.</td>
</tr>
<tr>
<td><code>#define LCD_COLOR_YELLOW</code></td>
<td>((uint32_t) 0xFFFFFF00)</td>
<td>Yellow value in ARGB8888 format.</td>
</tr>
<tr>
<td><code>#define LCD_COLOR_LIGHTBLUE</code></td>
<td>((uint32_t) 0xFF8080FF)</td>
<td>Light Blue value in ARGB8888 format.</td>
</tr>
<tr>
<td><code>#define LCD_COLOR_LIGHTGREEN</code></td>
<td>((uint32_t) 0xFF80FF80)</td>
<td>Light Green value in ARGB8888 format.</td>
</tr>
<tr>
<td><code>#define LCD_COLOR_LIGHTRED</code></td>
<td>((uint32_t) 0xFFFF8080)</td>
<td>Light Red value in ARGB8888 format.</td>
</tr>
<tr>
<td><code>#define LCD_COLOR_LIGHTCYAN</code></td>
<td>((uint32_t) 0xFF80FFFF)</td>
<td>Light Cyan value in ARGB8888 format.</td>
</tr>
<tr>
<td><code>#define LCD_COLOR_LIGHTMAGENTA</code></td>
<td>((uint32_t) 0xFFFF80FF)</td>
<td>Light Magenta value in ARGB8888 format.</td>
</tr>
<tr>
<td><code>#define LCD_COLOR_LIGHTYELLOW</code></td>
<td>((uint32_t) 0xFFFFFF80)</td>
<td>Light Yellow value in ARGB8888 format.</td>
</tr>
<tr>
<td><code>#define LCD_COLOR_DARKBLUE</code></td>
<td>((uint32_t) 0xFF000080)</td>
<td>Dark Blue value in ARGB8888 format.</td>
</tr>
<tr>
<td><code>#define LCD_COLOR_DARKGREEN</code></td>
<td>((uint32_t) 0xFF008000)</td>
<td>Light Dark Green value in ARGB8888 format.</td>
</tr>
<tr>
<td><code>#define LCD_COLOR_DARKRED</code></td>
<td>((uint32_t) 0xFF800000)</td>
<td>Light Dark Red value in ARGB8888 format.</td>
</tr>
<tr>
<td><code>#define LCD_COLOR_DARKCYAN</code></td>
<td>((uint32_t) 0xFF008080)</td>
<td>Dark Cyan value in ARGB8888 format.</td>
</tr>
<tr>
<td><code>#define LCD_COLOR_DARKMAGENTA</code></td>
<td>((uint32_t) 0xFF800080)</td>
<td>Dark Magenta value in ARGB8888 format.</td>
</tr>
</tbody>
</table>
#define **LCD_COLOR_DARKYELLOW** ((uint32_t) 0xFF808000)  
Dark Yellow value in ARGB8888 format.

#define **LCD_COLOR_WHITE** ((uint32_t) 0xFFFFFFFF)  
White value in ARGB8888 format.

#define **LCD_COLOR_LIGHTGRAY** ((uint32_t) 0xFFD3D3D3)  
Light Gray value in ARGB8888 format.

#define **LCD_COLOR_GRAY** ((uint32_t) 0xFF808080)  
Gray value in ARGB8888 format.

#define **LCD_COLOR_DARKGRAY** ((uint32_t) 0xFF404040)  
Dark Gray value in ARGB8888 format.

#define **LCD_COLOR_BLACK** ((uint32_t) 0xFF000000)  
Black value in ARGB8888 format.

#define **LCD_COLOR_BROWN** ((uint32_t) 0xFFA52A2A)  
Brown value in ARGB8888 format.

#define **LCD_COLOR_ORANGE** ((uint32_t) 0xFFFFA500)  
Orange value in ARGB8888 format.

#define **LCD_COLOR_TRANSPARENT** ((uint32_t) 0xFF000000)  
Transparent value in ARGB8888 format.

#define **LCD_DEFAULT_FONT** Font24  
LCD default font.

#define **LCD_DSI_PIXEL_DATA_FMT_RBG888** DSI_RGB888  
Possible values of pixel data format (ie color coding) transmitted lane in DSI packets.

#define **LCD_DSI_PIXEL_DATA_FMT_RBG565** DSI_RGB565
## Define Documentation

```c
#define BSP_LCD_DMA2D_IRQHandler DMA2D_IRQHandler
```
Definition at line **78** of file `stm32f769i_discovery_lcd.h`.

```c
#define BSP_LCD_DSI_IRQHandler DSI_IRQHandler
```
Definition at line **79** of file `stm32f769i_discovery_lcd.h`.

```c
#define BSP_LCD_LTDC_ER_IRQHandler LTDC_ER_IRQHandler
```
Definition at line **81** of file `stm32f769i_discovery_lcd.h`.

```c
#define BSP_LCD_LTDC_IRQHandler LTDC_IRQHandler
```
Definition at line **80** of file `stm32f769i_discovery_lcd.h`.

```c
#define HDMI_ADV7533_ID ((uint32_t) 0)
```
HDMI ADV7533 DSI Virtual Channel ID.
Definition at line **125** of file `stm32f769i_discovery_lcd.h`.

```c
#define HDMI_FORMAT_720_480 ((uint8_t) 0x00) /*720_480 format choice of HDMI display*/
```
HDMI Format.
Definition at line **130** of file `stm32f769i_discovery_lcd.h`.
```c
#define HDMI_FORMAT_720_576 ((uint8_t)0x01) /* 720_576 format */

Definition at line 131 of file stm32f769i_discovery_lcd.h.
Referenced by BSP_LCD_InitEx().

#define LCD_COLOR_BLACK ((uint32_t)0xFF000000)

Black value in ARGB8888 format.
Definition at line 228 of file stm32f769i_discovery_lcd.h.
Referenced by BSP_LCD_LayerDefaultInit().

#define LCD_COLOR_BLUE ((uint32_t)0xFF0000FF)

LCD color definitions values in ARGB8888 format.
Blue value in ARGB8888 format
Definition at line 140 of file stm32f769i_discovery_lcd.h.

#define LCD_COLOR_BROWN ((uint32_t)0xFFA52A2A)

Brown value in ARGB8888 format.
Definition at line 232 of file stm32f769i_discovery_lcd.h.

#define LCD_COLOR_CYAN ((uint32_t)0xFF00FFFF)

Cyan value in ARGB8888 format.
Definition at line 152 of file stm32f769i_discovery_lcd.h.
```
<table>
<thead>
<tr>
<th>Defines</th>
<th>Value in ARGB8888 format</th>
<th>Definition Line</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#define LCD_COLOR_DARKBLUE</code></td>
<td>0xFF000080</td>
<td>188</td>
<td><code>stm32f769i_discovery_lcd.h</code></td>
</tr>
<tr>
<td><code>#define LCD_COLOR_DARKCYAN</code></td>
<td>0xFF008080</td>
<td>200</td>
<td><code>stm32f769i_discovery_lcd.h</code></td>
</tr>
<tr>
<td><code>#define LCD_COLOR_DARKGRAY</code></td>
<td>0xFF404040</td>
<td>224</td>
<td><code>stm32f769i_discovery_lcd.h</code></td>
</tr>
<tr>
<td><code>#define LCD_COLOR_DARKGREEN</code></td>
<td>0xFF008000</td>
<td>192</td>
<td><code>stm32f769i_discovery_lcd.h</code></td>
</tr>
<tr>
<td><code>#define LCD_COLOR_DARKMAGENTA</code></td>
<td>0xFF800080</td>
<td>204</td>
<td><code>stm32f769i_discovery_lcd.h</code></td>
</tr>
<tr>
<td><code>#define LCD_COLOR_DARKRED</code></td>
<td>0xFF800000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Light Dark Red value in ARGB8888 format.
Definition at line 196 of file stm32f769i_discovery_lcd.h.

#define LCD_COLOR_DARKYELLOW ((uint32_t) 0xFF808000)

Dark Yellow value in ARGB8888 format.
Definition at line 208 of file stm32f769i_discovery_lcd.h.

#define LCD_COLOR_GRAY ((uint32_t) 0xFF808080)

Gray value in ARGB8888 format.
Definition at line 220 of file stm32f769i_discovery_lcd.h.

#define LCD_COLOR_GREEN ((uint32_t) 0xFF00FF00)

Green value in ARGB8888 format.
Definition at line 144 of file stm32f769i_discovery_lcd.h.

#define LCD_COLOR_LIGHTBLUE ((uint32_t) 0xFF8080FF)

Light Blue value in ARGB8888 format.
Definition at line 164 of file stm32f769i_discovery_lcd.h.

#define LCD_COLOR_LIGHTCYAN ((uint32_t) 0xFF80FFFF)

Light Cyan value in ARGB8888 format.
Definition at line 176 of file `stm32f769i_discovery_lcd.h`.

```c
#define LCD_COLOR_LIGHTGRAY ((uint32_t) 0xFFD3D3D3)
```

Light Gray value in ARGB8888 format.

Definition at line 216 of file `stm32f769i_discovery_lcd.h`.

```c
#define LCD_COLOR_LIGHTGREEN ((uint32_t) 0xFF80FF80)
```

Light Green value in ARGB8888 format.

Definition at line 168 of file `stm32f769i_discovery_lcd.h`.

```c
#define LCD_COLOR_LIGHTMAGENTA ((uint32_t) 0xFFFF80FF)
```

Light Magenta value in ARGB8888 format.

Definition at line 180 of file `stm32f769i_discovery_lcd.h`.

```c
#define LCD_COLOR_LIGHTRED ((uint32_t) 0xFFFF8080)
```

Light Red value in ARGB8888 format.

Definition at line 172 of file `stm32f769i_discovery_lcd.h`.

```c
#define LCD_COLOR_LIGHTYELLOW ((uint32_t) 0xFFFFFF80)
```

Light Yellow value in ARGB8888 format.

Definition at line 184 of file `stm32f769i_discovery_lcd.h`. 
#define LCD_COLOR_MAGENTA ((uint32_t) 0xFFFF00FF)
Magenta value in ARGB8888 format.
Definition at line 156 of file stm32f769i_discovery_lcd.h.

#define LCD_COLOR_ORANGE ((uint32_t) 0xFFFFA500)
Orange value in ARGB8888 format.
Definition at line 236 of file stm32f769i_discovery_lcd.h.

#define LCD_COLOR_RED ((uint32_t) 0xFFFF0000)
Red value in ARGB8888 format.
Definition at line 148 of file stm32f769i_discovery_lcd.h.

#define LCD_COLOR_TRANSPARENT ((uint32_t) 0xFF000000)
Transparent value in ARGB8888 format.
Definition at line 240 of file stm32f769i_discovery_lcd.h.

#define LCD_COLOR_WHITE ((uint32_t) 0xFFFFFFFF)
White value in ARGB8888 format.
Definition at line 212 of file stm32f769i_discovery_lcd.h.
Referenced by BSP_LCD_LayerDefaultInit().

#define LCD_COLOR_YELLOW ((uint32_t) 0xFFFFFF00)
Yellow value in ARGB888 format.
Definition at line 160 of file stm32f769i_discovery_lcd.h.

#define LCD_DEFAULT_FONT Font24

LCD default font.
Definition at line 245 of file stm32f769i_discovery_lcd.h.
Referenced by BSP_LCD_InitEx().

#define LCD_DSI_PIXEL_DATA_FMT_RBG565 DSI_RGB565

DSI packet pixel format chosen is RGB565 : 16 bpp
Definition at line 253 of file stm32f769i_discovery_lcd.h.

#define LCD_DSI_PIXEL_DATA_FMT_RBG888 DSI_RGB888

Possible values of pixel data format (ie color coding) transmitted on DSI Data lane in DSI packets.
DSI packet pixel format chosen is RGB888 : 24 bpp
Definition at line 252 of file stm32f769i_discovery_lcd.h.
Referenced by BSP_LCD_InitEx().

#define LCD_ERROR 0x01

Definition at line 114 of file stm32f769i_discovery_lcd.h.
Referenced by `BSP_LCD_InitEx()`.

```c
#define LCD_FB_START_ADDRESS ((uint32_t)0xC0000000)
```

LCD FB_StartAddress.

Definition at line 88 of file `stm32f769i_discovery_lcd.h`.

```c
#define LCD_LayerCfgTypeDef LTDC_LayerCfgTypeDef
```

Definition at line 84 of file `stm32f769i_discovery_lcd.h`.

Referenced by `BSP_LCD_LayerDefaultInit()`.

```c
#define LCD_OK 0x00
```

LCD status structure definition.

Definition at line 113 of file `stm32f769i_discovery_lcd.h`.

Referenced by `BSP_LCD_InitEx()`.

```c
#define LCD_OTM8009A_ID ((uint32_t)0)
```

LCD Display OTM8009A DSI Virtual Channel ID.

Definition at line 120 of file `stm32f769i_discovery_lcd.h`.

Referenced by `BSP_LCD_InitEx()`, `BSP_LCD_SetBrightness()`, and `DSI_IO_WriteCmd()`.

```c
#define LCD_TIMEOUT 0x02
```
#define LTDC_ACTIVE_LAYER_BACKGROUND  ((uint32_t) 0)

LTDC Background layer index.

Definition at line 115 of file stm32f769i_discovery_lcd.h.

#define LTDC_ACTIVE_LAYER_FOREGROUND  ((uint32_t) 1)

LTDC Foreground layer index.

Definition at line 96 of file stm32f769i_discovery_lcd.h.

#define LTDC_DEFAULT_ACTIVE_LAYER  LTDC_ACTIVE_LAYER_FOREGROUND

LTDC Default used layer index.

Definition at line 100 of file stm32f769i_discovery_lcd.h.

#define LTDC_MAX_LAYER_NUMBER  ((uint32_t) 2)

Maximum number of LTDC layers.

Definition at line 92 of file stm32f769i_discovery_lcd.h.

#define LTDC_NB_OF_LAYERS  ((uint32_t) 2)

Number of LTDC layers.

Definition at line 104 of file stm32f769i_discovery_lcd.h.
### STM32F769I_DISCOVERY QSPI Exported Functions

- STM32F769I_DISCOVERY_QSPI
## Functions

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

`uint8_t BSP_QSPI_DeInit ( void )`

De-Initializes the QSPI interface.

**Return values:**
- `QSPI` memory status

Definition at line 188 of file `stm32f769i_discovery_qspi.c`.

References `BSP_QSPI_MspDeInit()`, `QSPI_ERROR`, `QSPI_ExitMemory_QPI()`, `QSPI_NOT_SUPPORTED`, `QSPI_OK`, and `QSPIHandle`.

`uint8_t BSP_QSPI_EnableMemoryMappedMode ( void )`

Configure the QSPI in memory-mapped mode.

**Return values:**
- `QSPI` memory status

Definition at line 480 of file `stm32f769i_discovery_qspi.c`.

References `QSPI_ERROR`, `QSPI_OK`, and `QSPIHandle`.

`uint8_t BSP_QSPI_Erase_Block ( uint32_t BlockAddress )`

Erases the specified block of the QSPI memory.

**Parameters:**
- `BlockAddress`: Block address to erase

**Return values:**
**QSPI memory status**

Definition at line 337 of file `stm32f769i_discovery_qspi.c`.

References `QSPI_AutoPollingMemReady()`, `QSPI_ERROR`, `QSPI_OK`, `QSPI_WriteEnable()`, and `QSPIHandle`.

```c
uint8_t BSP_QSPI_Erase_Chip ( void )
```

Erases the entire QSPI memory.

**Return values:**

**QSPI memory status**

Definition at line 379 of file `stm32f769i_discovery_qspi.c`.

References `QSPI_AutoPollingMemReady()`, `QSPI_ERROR`, `QSPI_OK`, `QSPI_WriteEnable()`, and `QSPIHandle`.

```c
uint8_t BSP_QSPI_GetInfo ( QSPI_Info * pInfo )
```

Return the configuration of the QSPI memory.

**Parameters:**

- `pInfo`: pointer on the configuration structure

**Return values:**

**QSPI memory status**

Definition at line 464 of file `stm32f769i_discovery_qspi.c`.

References `QSPI_Info::EraseSectorSize`, `QSPI_Info::EraseSectorsNumber`, `QSPI_Info::FlashSize`, `QSPI_Info::ProgPageSize`, `QSPI_Info::ProgPagesNumber`, and `QSPI_OK`. 

```c
```
### uint8_t BSP_QSPI_GetStatus ( void )

Reads current status of the QSPI memory.

**Return values:**
- QSPI memory status

Definition at line 419 of file *stm32f769i_discovery_qspi.c*.

References QSPI_BUSY, QSPI_ERROR, QSPI_OK, and QSPIHandle.

### uint8_t BSP_QSPI_Init ( void )

Initializes the QSPI interface.

**Return values:**
- QSPI memory status

Definition at line 122 of file *stm32f769i_discovery_qspi.c*.

References BSP_QSPI_MspInit(), QSPI_DummyCyclesCfg(), QSPI_EnterFourBytesAddress(), QSPI_EnterMemory_QPI(), QSPI_ERROR, QSPI_NOT_SUPPORTED, QSPI_OK, QSPI_OutDrvStrengthCfg(), QSPI_ResetMemory(), and QSPIHandle.

### void BSP_QSPI_MspDeInit ( QSPI_HandleTypeDef * hqspi, void * Params )

QSPI MSP De-Initialization This function frees the hardware resources used in this example:
• Disable the Peripheral's clock
• Revert GPIO and NVIC configuration to their default state

**Return values:**

None

Definition at line 586 of file `stm32f769i_discovery_qspi.c`.

References `QSPI_CLK_DISABLE`, `QSPI_CLK_GPIO_PORT`, `QSPI_CLK_PIN`, `QSPI_CS_GPIO_PORT`, `QSPI_CS_PIN`, `QSPI_D0_GPIO_PORT`, `QSPI_D0_PIN`, `QSPI_D1_GPIO_PORT`, `QSPI_D1_PIN`, `QSPI_D2_GPIO_PORT`, `QSPI_D2_PIN`, `QSPI_D3_GPIO_PORT`, `QSPI_D3_PIN`, `QSPI_FORCE_RESET`, and `QSPI_RELEASE_RESET`.

Referenced by `BSP_QSPI_DeInit()`.

```c
void BSP_QSPI_MspInit ( QSPI_HandleTypeDef * hqspi,
                                   void * Params
)
```

QSPI MSP Initialization This function configures the hardware resources used in this example:

• Peripheral's clock enable
• Peripheral's GPIO Configuration
• NVIC configuration for QSPI interrupt

**Return values:**

None

Definition at line 525 of file `stm32f769i_discovery_qspi.c`.

References `QSPI_CLK_ENABLE`,
`QSPI_CLK_GPIO_CLK_ENABLE`, `QSPI_CLK_GPIO_PORT`,
`QSPI_CLK_PIN`, `QSPI_CLK_PIN_AF`,
`QSPI_CS_GPIO_CLK_ENABLE`, `QSPI_CS_GPIO_PORT`,
`QSPI_CS_PIN`, `QSPI_CS_PIN_AF`, `QSPI_D0_GPIO_CLK_ENABLE`,
`QSPI_D1_GPIO_CLK_ENABLE`, `QSPI_D1_GPIO_PORT`,
`QSPI_D1_PIN`, `QSPI_D2_GPIO_PORT`, `QSPI_D2_PIN`,
`QSPI_D2_PIN`, `QSPI_D3_GPIO_PORT`, `QSPI_D3_PIN`, `QSPI_FORCE_RESET`, and `QSPI_RELEASE_RESET`. 
QSPI_D0_GPIO_PORT, QSPI_D0_PIN, QSPI_D0_PIN_AF,
QSPI_D1_GPIO_CLK_ENABLE, QSPI_D1_GPIO_PORT,
QSPI_D1_PIN, QSPI_D1_PIN_AF, QSPI_D2_GPIO_CLK_ENABLE,
QSPI_D2_GPIO_PORT, QSPI_D2_PIN, QSPI_D2_PIN_AF,
QSPI_D3_GPIO_CLK_ENABLE, QSPI_D3_GPIO_PORT,
QSPI_D3_PIN, QSPI_D3_PIN_AF, QSPI_FORCE_RESET, and
QSPI_RELEASE_RESET.

Referenced by **BSP_QSPI_Init**().

```c
uint8_t BSP_QSPI_Read ( uint8_t * pData,
    uint32_t ReadAddr,
    uint32_t Size
  )
```

Reads an amount of data from the QSPI memory.

**Parameters:**
- **pData:** Pointer to data to be read
- **ReadAddr:** Read start address
- **Size:** Size of data to read

**Return values:**
- **QSPI** memory status

Definition at line 217 of file `stm32f769i_discovery_qspi.c`.

References **QSPI_ERROR**, **QSPI_OK**, and **QSPIHandle**.

```c
uint8_t BSP_QSPI_Write ( uint8_t * pData,
    uint32_t WriteAddr,
    uint32_t Size
  )
```
Writes an amount of data to the QSPI memory.

**Parameters:**
- **pData:** Pointer to data to be written
- **WriteAddr:** Write start address
- **Size:** Size of data to write

**Return values:**
- QSPI memory status

Definition at line 263 of file `stm32f769i_discovery_qspi.c`.

References `QSPI_AutoPollingMemReady()`, `QSPI_ERROR`, `QSPI_OK`, `QSPI_WriteEnable()`, and `QSPIHandle`.

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by [doxygen](http://www.doxygen.org) 1.7.6.1
## STM32F769I-Discovery BSP User Manual

###Directories

**STM32F769I_DISCOVERY QSPI Private Functions**

**STM32F769I_DISCOVERY QSPI**
### Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static uint8_t QSPI_ResetMemory(uint8_t *hqspi)</code></td>
<td>This function resets the QSPI memory.</td>
</tr>
<tr>
<td><code>static uint8_t QSPI_EnterFourBytesAddress(uint8_t *hqspi)</code></td>
<td>This function sets the QSPI memory in 4-byte address mode.</td>
</tr>
<tr>
<td><code>static uint8_t QSPI_DummyCyclesCfg(uint8_t *hqspi)</code></td>
<td>This function configures the dummy cycles on the memory side.</td>
</tr>
<tr>
<td><code>static uint8_t QSPI_EnterMemory_QPI(uint8_t *hqspi)</code></td>
<td>This function puts the QSPI memory in QPI mode (quad I/O).</td>
</tr>
<tr>
<td><code>static uint8_t QSPI_ExitMemory_QPI(uint8_t *hqspi)</code></td>
<td>This function puts the QSPI memory in SPI mode.</td>
</tr>
<tr>
<td><code>static uint8_t QSPI_OutDrvStrengthCfg(uint8_t *hqspi)</code></td>
<td>This function configures the Output driver strength on the memory side.</td>
</tr>
<tr>
<td><code>static uint8_t QSPI_WriteEnable(uint8_t *hqspi)</code></td>
<td>This function sends a Write Enable and waits for it to be effective.</td>
</tr>
<tr>
<td><code>static uint8_t QSPI_AutoPollingMemReady(uint8_t *hqspi, uint32_t Timeout)</code></td>
<td>This function reads the SR of the memory and waits for the EOP.</td>
</tr>
<tr>
<td><code>__weak void BSP_QSPI_MspInit(uint8_t *hqspi)</code></td>
<td>QSPI MSP Initialization This function configures the hardware resources used in this example:</td>
</tr>
<tr>
<td><code>BSP_QSPI_MspDeInit(uint8_t *hqspi)</code></td>
<td>QSPI MSP Deinitialization This function deinitializes the hardware resources used in this example:</td>
</tr>
</tbody>
</table>
__weak void *Params)

QSPI MSP De-Initialization

This function frees the

hardware resources used in this example:
__weak void BSP_QSPI_MspDeInit (QSPI_HandleTypeDef * hqspi,
void * Param)

QSPI MSP De-Initialization This function frees the hardware resources used in this example:

- Disable the Peripheral's clock
- Revert GPIO and NVIC configuration to their default state

Return values:
None

Definition at line 586 of file stm32f769i_discovery_qspi.c.

References QSPI_CLK_DISABLE, QSPI_CLK_GPIO_PORT, QSPI_CLK_PIN, QSPI_CS_GPIO_PORT, QSPI_CS_PIN, QSPI_D0_GPIO_PORT, QSPI_D0_PIN, QSPI_D1_GPIO_PORT, QSPI_D1_PIN, QSPI_D2_GPIO_PORT, QSPI_D2_PIN, QSPI_D3_GPIO_PORT, QSPI_D3_PIN, QSPI_FORCE_RESET, and QSPI_RELEASE_RESET.

Referenced by BSP_QSPI_DeInit().

__weak void BSP_QSPI_MspInit (QSPI_HandleTypeDef * hqspi,
void * Param)

QSPI MSP Initialization This function configures the hardware resources used in this example:

- Peripheral's clock enable
- Peripheral's GPIO Configuration
• NVIC configuration for QSPI interrupt

**Return values:**

None

Definition at line 525 of file stm32f769i_discovery_qspi.c.

References `QSPI_CLK_ENABLE`, `QSPI_CLK_GPIO_CLK_ENABLE`, `QSPI_CLK_GPIO_PORT`, `QSPI_CLK_PIN`, `QSPI_CLK_PIN_AF`, `QSPI_CS_GPIO_CLK_ENABLE`, `QSPI_CS_GPIO_PORT`, `QSPI_CS_PIN`, `QSPI_CS_PIN_AF`, `QSPI_D0_GPIO_CLK_ENABLE`, `QSPI_D0_GPIO_PORT`, `QSPI_D0_PIN`, `QSPI_D0_PIN_AF`, `QSPI_D1_GPIO_CLK_ENABLE`, `QSPI_D1_GPIO_PORT`, `QSPI_D1_PIN`, `QSPI_D1_PIN_AF`, `QSPI_D2_GPIO_CLK_ENABLE`, `QSPI_D2_GPIO_PORT`, `QSPI_D2_PIN`, `QSPI_D2_PIN_AF`, `QSPI_D3_GPIO_CLK_ENABLE`, `QSPI_D3_GPIO_PORT`, `QSPI_D3_PIN`, `QSPI_D3_PIN_AF`, `QSPI_FORCE_RESET`, and `QSPI_RELEASE_RESET`.

Referenced by `BSP_QSPI_Init()`.

```c
static uint8_t QSPI_AutoPollingMemReady ( QSPI_HandleTypeDef *hqspi, uint32_t Timeout )
```

This function read the SR of the memory and wait the EOP.

**Parameters:**

- `hqspi`: QSPI handle
- `Timeout`

**Return values:**

None

Definition at line 1130 of file stm32f769i_discovery_qspi.c.
References **QSPI_ERROR**, and **QSPI_OK**.

Referenced by **BSP_QSPI_Erase_Block()**, **BSP_QSPI_Erase_Chip()**, **BSP_QSPI_Write()**, and **QSPI_EnterFourBytesAddress()**.

```c
static uint8_t QSPI_DummyCyclesCfg (QSPI_HandleTypeDef *hqspi)
```

This function configure the dummy cycles on memory side.

**Parameters:**
- `hqspi`: QSPI handle

**Return values:**
- `None`

Definition at line 815 of file **stm32f769i_discovery_qspi.c**.

References **QSPI_ERROR**, **QSPI_OK**, and **QSPI_WriteEnable()**.

Referenced by **BSP_QSPI_Init()**.

```c
static uint8_t QSPI_EnterFourBytesAddress (QSPI_HandleTypeDef *hqspi)
```

This function set the QSPI memory in 4-byte address mode.

**Parameters:**
- `hqspi`: QSPI handle

**Return values:**
- `None`

Definition at line 774 of file **stm32f769i_discovery_qspi.c**.

References **QSPI_AutoPollingMemReady()**, **QSPI_ERROR**, **QSPI_OK**, and **QSPI_WriteEnable()**.
static uint8_t QSPI_EnterMemory_QPI (QSPI_HandleTypeDef *hqspi)

This function put QSPI memory in QPI mode (quad I/O).

**Parameters:**
- **hqspi:** QSPI handle

**Return values:**
- **None**

Definition at line 912 of file stm32f769i_discovery_qspi.c.

References **QSPI_ERROR**, and **QSPI_OK**.

Referenced by **BSP_QSPI_Init()**.

static uint8_t QSPI_ExitMemory_QPI (QSPI_HandleTypeDef *hqspi)

This function put QSPI memory in SPI mode.

**Parameters:**
- **hqspi:** QSPI handle

**Return values:**
- **None**

Definition at line 960 of file stm32f769i_discovery_qspi.c.

References **QSPI_ERROR**, and **QSPI_OK**.

Referenced by **BSP_QSPI_DeInit()**.
static uint8_t QSPI_OutDrvStrengthCfg (QSPI_HandleTypeDef * hqspi)

This function configure the Output driver strength on memory side.

**Parameters:**
- *hqspi:* QSPI handle

**Return values:**
- None

Definition at line 990 of file stm32f769i_discovery_qspi.c.

References QSPI_ERROR, QSPI_OK, QSPI_WriteEnable(), and QSPIHandle.

Referenced by BSP_QSPI_Init().

---

static uint8_t QSPI_ResetMemory (QSPI_HandleTypeDef * hqspi)

This function reset the QSPI memory.

**Parameters:**
- *hqspi:* QSPI handle

**Return values:**
- None

Definition at line 614 of file stm32f769i_discovery_qspi.c.

References QSPI_ERROR, and QSPI_OK.

Referenced by BSP_QSPI_Init().

---

static uint8_t QSPI_WriteEnable (QSPI_HandleTypeDef * hqspi)

---
This function sends a Write Enable and wait it is effective.

**Parameters:**

- `hqspi`: QSPI handle

**Return values:**

- `None`

Definition at line 1084 of file `stm32f769i_discovery_qspi.c`.

References `QSPI_ERROR`, and `QSPI_OK`.

Referenced by `BSP_QSPI_Erase_Block()`, `BSP_QSPI_Erase_Chip()`, `BSP_QSPI_Write()`, `QSPI_DummyCyclesCfg()`, `QSPI_EnterFourBytesAddress()`, and `QSPI_OutDrvStrengthCfg()`.

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery
BSP User Manual by [doxygen](http://www.stackoverflow.com) 1.7.6.1
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**STM32F769I Discovery**  
**Sd Private Functions**

STM32F769I_DISCOVERY_SD
## Functions

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_Init</strong> (void)</td>
<td>Initializes the SD card device.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_DeInit</strong> (void)</td>
<td>DeInitializes the SD card device.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_ITConfig</strong> (void)</td>
<td>Configures Interrupt mode for SD detection pin.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_IsDetected</strong> (void)</td>
<td>Detects if SD card is correctly plugged in the memory slot or not.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_ReadBlocks</strong> (uint32_t *pData, uint32_t ReadAddr, uint32_t NumOfBlocks, uint32_t Timeout)</td>
<td>Reads block(s) from a specified address in an SD card, in polling mode.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_WriteBlocks</strong> (uint32_t *pData, uint32_t WriteAddr, uint32_t NumOfBlocks, uint32_t Timeout)</td>
<td>Writes block(s) to a specified address in an SD card, in polling mode.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_ReadBlocks_DMA</strong> (uint32_t *pData, uint32_t ReadAddr, uint32_t NumOfBlocks)</td>
<td>Reads block(s) from a specified address in an SD card, in DMA mode.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_WriteBlocks_DMA</strong> (uint32_t *pData, uint32_t WriteAddr, uint32_t NumOfBlocks)</td>
<td>Writes block(s) to a specified address in an SD card, in DMA mode.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_SD_Erase</strong> (uint32_t StartAddr, uint32_t EndAddr)</td>
<td>Erases the specified memory area of the given SD card.</td>
</tr>
<tr>
<td>__weak</td>
<td><strong>BSP_SD_MspInit</strong> (SD_HandleTypeDef *hsd, void *Params)</td>
<td>Initializes the SD MSP.</td>
</tr>
<tr>
<td>Function Name</td>
<td>Arguments</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>__weak void BSP_SD_Detect_MspInit</td>
<td>(SD_HandleTypeDef *hsd, void *Params)</td>
<td>Initializes the SD Detect pin MSP.</td>
</tr>
<tr>
<td>__weak void BSP_SD_MspDeInit</td>
<td>(SD_HandleTypeDef *hsd, void *Params)</td>
<td>DeInitializes the SD MSP.</td>
</tr>
<tr>
<td>uint8_t BSP_SD_GetCardState</td>
<td>(void)</td>
<td>Gets the current SD card data status.</td>
</tr>
<tr>
<td>void BSP_SD_GetCardInfo</td>
<td>(HAL_SD_CardInfoTypeDef *CardInfo)</td>
<td>Get SD information about specific SD card.</td>
</tr>
<tr>
<td>void HAL_SD_AbortCallback</td>
<td>(SD_HandleTypeDef *hsd)</td>
<td>SD Abort callbacks.</td>
</tr>
<tr>
<td>void HAL_SD_TxCpltCallback</td>
<td>(SD_HandleTypeDef *hsd)</td>
<td>Tx Transfer completed callbacks.</td>
</tr>
<tr>
<td>void HAL_SD_RxCpltCallback</td>
<td>(SD_HandleTypeDef *hsd)</td>
<td>Rx Transfer completed callbacks.</td>
</tr>
<tr>
<td>__weak void BSP_SD_AbortCallback</td>
<td>(void)</td>
<td>BSP SD Abort callbacks.</td>
</tr>
<tr>
<td>__weak void BSP_SD_WriteCpltCallback</td>
<td>(void)</td>
<td>BSP Tx Transfer completed callbacks.</td>
</tr>
<tr>
<td>__weak void BSP_SD_ReadCpltCallback</td>
<td>(void)</td>
<td>BSP Rx Transfer completed callbacks.</td>
</tr>
</tbody>
</table>
**Function Documentation**

```c
__weak void BSP_SD_AbortCallback ( void )
```

BSP SD Abort callbacks.

**Return values:**

*None*

Definition at line 565 of file `stm32f769i_discovery_sd.c`.

Referenced by `HAL_SD_AbortCallback()`.

```c
uint8_t BSP_SD_DeInit ( void )
```

DeInitializes the SD card device.

**Return values:**

*SD* status

Definition at line 192 of file `stm32f769i_discovery_sd.c`.

References `BSP_SD_MspDeInit()`, `MSD_ERROR`, `MSD_OK`, and `uSdHandle`.

```c
__weak void BSP_SD_Detect_MspInit ( SD_HandleTypeDef * hsd, void * Params )
```

Initializes the SD Detect pin MSP.

**Parameters:**

*hsd,:* SD handle
**Params**: pointer on additional configuration parameters, can be NULL.

**Return values:**

None

Definition at line 458 of file `stm32f769i_discovery_sd.c`.

References `SD_DETECT_GPIO_CLK_ENABLE`, `SD_DETECT_GPIO_PORT`, and `SD_DETECT_PIN`.

Referenced by `BSP_SD_Init()`.

```c
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 336 of file `stm32f769i_discovery_sd.c`.

References `MSD_ERROR`, `MSD_OK`, and `uSdHandle`.

```c
void BSP_SD_GetCardInfo ( HAL_SD_CardInfoTypeDef * CardInfo )
```

Get SD information about specific SD card.

**Parameters:**
**CardInfo:** Pointer to HAL_SD_CardInfoTypeDef structure

**Return values:**
None

Definition at line 525 of file stm32f769i_discovery_sd.c.

References **uSdHandle**.

```c
uint8_t BSP_SD_GetCardState ( void )
```

Gets the current SD card data status.

**Return values:**

- **Data** transfer state. This value can be one of the following values:
  - SD_TRANSFER_OK: No data transfer is acting
  - SD_TRANSFER_BUSY: Data transfer is acting

Definition at line 514 of file stm32f769i_discovery_sd.c.

References **SD_TRANSFER_BUSY, SD_TRANSFER_OK**, and **uSdHandle**.

```c
uint8_t BSP_SD_Init ( void )
```

Initializes the SD card device.

**Return values:**

- **SD** status

Definition at line 141 of file stm32f769i_discovery_sd.c.

References **BSP_SD_Detect_MspInit(), BSP_SD_IsDetected(), BSP_SD_MspInit(), MSD_ERROR**,
### 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 237 of file `stm32f769i_discovery_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_ITConfig ( void )

Configures Interrupt mode for SD detection pin.

**Return values:**

- **Returns** 0

Definition at line 215 of file `stm32f769i_discovery_sd.c`.

References `MSD_OK`, `SD_DETECT_EXTI_IRQn`, `SD_DETECT_GPIO_PORT`, and `SD_DETECT_PIN`.

### __weak void BSP_SD_MspDeInit ( SD_HandleTypeDef * hsd, void * Params )

DeInitializes the SD MSP.
Parameters:

hsd,: SD handle

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

Definition at line 477 of file stm32f769i_discovery_sd.c.

References SD_DMAx_Rx_IRQn, SD_DMAx_Rx_STREAM, SD_DMAx.Tx_IRQn, and SD_DMAx.Tx_STREAM.

Referenced by BSP_SD_DeInit().

__weak void BSP_SD_MspInit ( SD_HandleTypeDef * hsd,
                              void * Params
                          )

Initializes the SD MSP.

Parameters:

hsd,: SD handle

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

Definition at line 353 of file stm32f769i_discovery_sd.c.

References __DMAx_TxRx_CLK_ENABLE, SD_DMAx_Rx_CHANNEL, SD_DMAx_Rx_IRQn, SD_DMAx_Rx_STREAM, SD_DMAx.Tx_CHANNEL, SD_DMAx.Tx_IRQn, and SD_DMAx.Tx_STREAM.

Referenced by BSP_SD_Init().

uint8_t BSP_SD_ReadBlocks ( uint32_t * pData,
                            uint32_t ReadAddr,
                            uint32_t NumOfBlocks,
uint32_t Timeout
)

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

**Parameters:**
- **pData:** Pointer to the buffer that will contain the data to transmit
- **ReadAddr:** Address from where data is to be read
- **NumOfBlocks:** Number of SD blocks to read
- **Timeout:** Timeout for read operation

**Return values:**
- **SD status**

Definition at line 258 of file `stm32f769i_discovery_sd.c`.

References **MSD_ERROR, MSD_OK**, and **uSdHandle**.

```c
uint8_t BSP_SD_ReadBlocks_DMA ( uint32_t * pData,
    uint32_t ReadAddr,
    uint32_t NumOfBlocks )
```

Reads block(s) from a specified address in an SD card, in DMA mode.

**Parameters:**
- **pData:** Pointer to the buffer that will contain the data to transmit
- **ReadAddr:** Address from where data is to be read
- **NumOfBlocks:** Number of SD blocks to read

**Return values:**
**SD status**

Definition at line 297 of file stm32f769i_discovery_sd.c.

References **MSD_ERROR, MSD_OK, and uSdHandle**.

```c
__weak void BSP_SD_ReadCpltCallback ( void )
```

BSP Rx Transfer completed callbacks.

**Return values:**

**None**

Definition at line 583 of file stm32f769i_discovery_sd.c.

Referenced by **HAL_SD_RxCpltCallback()**.

```c
uint8_t BSP_SD_WriteBlocks ( uint32_t * pData,
    uint32_t WriteAddr,
    uint32_t NumOfBlocks,
    uint32_t Timeout
)
```

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

**Parameters:**

- **pData:** Pointer to the buffer that will contain the data to transmit
- **WriteAddr:** Address from where data is to be written
- **NumOfBlocks:** Number of SD blocks to write
- **Timeout:** Timeout for write operation

**Return values:**

**SD** status
Definition at line 278 of file stm32f769i_discovery_sd.c.

References MSD_ERROR, MSD_OK, and uSdHandle.

```c
uint8_t BSP_SD_WriteBlocks_DMA ( uint32_t * pData,
                                uint32_t WriteAddr,
                                uint32_t NumOfBlocks )
```

Writes block(s) to a specified address in an SD card, in DMA mode.

**Parameters:**
- **pData:** Pointer to the buffer that will contain the data to transmit
- **WriteAddr:** Address from where data is to be written
- **NumOfBlocks:** Number of SD blocks to write

**Return values:**
- **SD status**

Definition at line 317 of file stm32f769i_discovery_sd.c.

References MSD_ERROR, MSD_OK, and uSdHandle.

```c
__weak void BSP_SD_WriteCpltCallback ( void )
```

BSP Tx Transfer completed callbacks.

**Return values:**
- **None**

Definition at line 574 of file stm32f769i_discovery_sd.c.

Referenced by HAL_SD_TxCpltCallback().
void HAL_SD_AbortCallback (SD_HandleTypeDef * hsd)

SD Abort callbacks.

**Parameters:**
- **hsd,:** SD handle

**Return values:**
- **None**

Definition at line 536 of file *stm32f769i_discovery_sd.c.*

References **BSP_SD_AbortCallback().**

---

void HAL_SD_RxCpltCallback (SD_HandleTypeDef * hsd)

Rx Transfer completed callbacks.

**Parameters:**
- **hsd,:** SD handle

**Return values:**
- **None**

Definition at line 556 of file *stm32f769i_discovery_sd.c.*

References **BSP_SD_ReadCpltCallback().**

---

void HAL_SD_TxCpltCallback (SD_HandleTypeDef * hsd)

Tx Transfer completed callbacks.

**Parameters:**
- **hsd,:** SD handle
Return values:

None

Definition at line 546 of file stm32f769i_discovery_sd.c.

References BSP_SD_WriteCpltCallback().
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td>Functions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

STM32F769I Discovery SD Exported Functions

STM32F769I_DISCOVERY SD
## Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BSP_SD_Init</code></td>
<td>(void)</td>
<td>Initializes the SD card device.</td>
</tr>
<tr>
<td><code>BSP_SD_DeInit</code></td>
<td>(void)</td>
<td>DeInitializes the SD card device.</td>
</tr>
<tr>
<td><code>BSP_SD_ITConfig</code></td>
<td>(void)</td>
<td>Configures Interrupt mode for SD detection pin.</td>
</tr>
<tr>
<td><code>BSP_SD_ReadBlocks</code></td>
<td>(uint32_t *pData, uint32_t ReadAddr, uint32_t NumOfBlocks, uint32_t Timeout)</td>
<td>Reads block(s) from a specified address in an SD card, in polling mode.</td>
</tr>
<tr>
<td><code>BSP_SD_WriteBlocks</code></td>
<td>(uint32_t *pData, uint32_t WriteAddr, uint32_t NumOfBlocks, uint32_t Timeout)</td>
<td>Writes block(s) to a specified address in an SD card, in polling mode.</td>
</tr>
<tr>
<td><code>BSP_SD_ReadBlocks_DMA</code></td>
<td>(uint32_t *pData, uint32_t ReadAddr, uint32_t NumOfBlocks)</td>
<td>Reads block(s) from a specified address in an SD card, in DMA mode.</td>
</tr>
<tr>
<td><code>BSP_SD_WriteBlocks_DMA</code></td>
<td>(uint32_t *pData, uint32_t WriteAddr, uint32_t NumOfBlocks)</td>
<td>Writes block(s) to a specified address in an SD card, in DMA mode.</td>
</tr>
<tr>
<td><code>BSP_SD_Erase</code></td>
<td>(uint32_t StartAddr, uint32_t EndAddr)</td>
<td>Erases the specified memory area of the given SD card.</td>
</tr>
<tr>
<td><code>BSP_SD_GetCardState</code></td>
<td>(void)</td>
<td>Gets the current SD card data status.</td>
</tr>
<tr>
<td><code>BSP_SD_GetCardInfo</code></td>
<td>(HAL_SD_CardInfoTypeDef *CardInfo)</td>
<td>Get SD information about specific SD card.</td>
</tr>
<tr>
<td><code>BSP_SD_IsDetected</code></td>
<td>(void)</td>
<td>Detects if SD card is correctly plugged in the memory slot or</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>void BSP_SD_MspInit (SD_HandleTypeDef *hsd, void *Params)</code></td>
<td>Initializes the SD MSP.</td>
<td></td>
</tr>
<tr>
<td><code>void BSP_SD_Detect_MspInit (SD_HandleTypeDef *hsd, void *Params)</code></td>
<td>Initializes the SD Detect pin MSP.</td>
<td></td>
</tr>
<tr>
<td><code>void BSP_SD_MspDeInit (SD_HandleTypeDef *hsd, void *Params)</code></td>
<td>DeInitializes the SD MSP.</td>
<td></td>
</tr>
<tr>
<td><code>void BSP_SD_AbortCallback (void)</code></td>
<td>BSP SD Abort callbacks.</td>
<td></td>
</tr>
<tr>
<td><code>void BSP_SD_WriteCpltCallback (void)</code></td>
<td>BSP Tx Transfer completed callbacks.</td>
<td></td>
</tr>
<tr>
<td><code>void BSP_SD_ReadCpltCallback (void)</code></td>
<td>BSP Rx Transfer completed callbacks.</td>
<td></td>
</tr>
</tbody>
</table>
Function Documentation

**void BSP_SD_AbortCallback ( void )**

BSP SD Abort callbacks.

**Return values:**

None

Definition at line 565 of file stm32f769i_discovery_sd.c.

Referenced by HAL_SD_AbortCallback().

**uint8_t BSP_SD_DeInit ( void )**

DeInitializes the SD card device.

**Return values:**

SD status

Definition at line 192 of file stm32f769i_discovery_sd.c.

References BSP_SD_MspDeInit(), MSD_ERROR, MSD_OK, and uSdHandle.

**void BSP_SD_Detect_MspInit ( SD_HandleTypeDef * hsd, void * Params )**

Initializes the SD Detect pin MSP.

**Parameters:**

hsd:, SD handle
**Params** : pointer on additional configuration parameters, can be NULL.

**Return values:**

*None*

Definition at line 458 of file *stm32f769i_discovery_sd.c*.

References **SD_DETECT_GPIO_CLK_ENABLE**, **SD_DETECT_GPIO_PORT**, and **SD_DETECT_PIN**.

Referenced by **BSP_SD_Init()**.

```c
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 336 of file *stm32f769i_discovery_sd.c*.

References **MSD_ERROR**, **MSD_OK**, and **uSdHandle**.

```c
void BSP_SD_GetCardInfo ( HAL_SD_CardInfoTypeDef * CardInfo )
```

Get SD information about specific SD card.

**Parameters:**
**CardInfo:** Pointer to `HAL_SD_CardInfoTypeDef` structure

**Return values:**

*None*

Definition at line 525 of file `stm32f769i_discovery_sd.c`.

References `uSdHandle`.

```c
uint8_t BSP_SD_GetCardState ( void )
```

Gets the current SD card data status.

**Return values:**

*Data* transfer state. This value can be one of the following values:

- `SD_TRANSFER_OK`: No data transfer is acting
- `SD_TRANSFER_BUSY`: Data transfer is acting

Definition at line 514 of file `stm32f769i_discovery_sd.c`.

References `SD_TRANSFER_BUSY`, `SD_TRANSFER_OK`, and `uSdHandle`.

```c
uint8_t BSP_SD_Init ( void )
```

Initializes the SD card device.

**Return values:**

*SD* status

Definition at line 141 of file `stm32f769i_discovery_sd.c`.

References `BSP_SD_Detect_MspInit()`, `BSP_SD_IsDetected()`, `BSP_SD_MspInit()`, `MSD_ERROR`,
MSD_ERROR_SD_NOT_PRESENT, MSD_OK, SD_PRESENT, and uSdHandle.

uint8_t BSP_SD_IsDetected ( void )

Detects if SD card is correctly plugged in the memory slot or not.

Return values:

Returns if SD is detected or not

Definition at line 237 of file stm32f769i_discovery_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_ITConfig ( void )

Configures Interrupt mode for SD detection pin.

Return values:

Returns 0

Definition at line 215 of file stm32f769i_discovery_sd.c.

References MSD_OK, SD_DETECT_EXTI_IRQn, SD_DETECT_GPIO_PORT, and SD_DETECT_PIN.

void BSP_SD_MspDeInit ( SD_HandleTypeDef * hsd,
                        void * Params )

DeInitializes the SD MSP.
Parameters:

hsd,: SD handle

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

Definition at line 477 of file stm32f769i_discovery_sd.c.

References SD_DMAx_Rx_IRQHandler, SD_DMAx_Rx_STREAM, SD_DMAx_Tx_IRQHandler, and SD_DMAx_Tx_STREAM.

Referenced by BSP_SD_DeInit().

```c
void BSP_SD_MspInit ( SD_HandleTypeDef * hsd,
                      void * Params
                  )
```

Initializes the SD MSP.

Parameters:

hsd,: SD handle

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

Definition at line 353 of file stm32f769i_discovery_sd.c.

References __DMAx_TxRx_CLK_ENABLE, SD_DMAx_Rx_CHANNEL, SD_DMAx_Rx_IRQHandler, SD_DMAx_Rx_STREAM, SD_DMAx_Tx_CHANNEL, SD_DMAx_Tx_IRQHandler, and SD_DMAx_Tx_STREAM.

Referenced by BSP_SD_Init().

```c
uint8_t BSP_SD_ReadBlocks ( uint32_t * pData,
                           uint32_t ReadAddr,
                           uint32_t NumOfBlocks,
                       )
```
uint32_t Timeout
)

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

Parameters:
   pData,: Pointer to the buffer that will contain the data to transmit
   ReadAddr,: Address from where data is to be read
   NumOfBlocks,: Number of SD blocks to read
   Timeout,: Timeout for read operation

Return values:
   SD status

Definition at line 258 of file stm32f769i_discovery_sd.c.

References MSD_ERROR, MSD_OK, and uSdHandle.

uint8_t BSP_SD_ReadBlocks_DMA ( uint32_t * pData,
   uint32_t ReadAddr,
   uint32_t NumOfBlocks
)

Reads block(s) from a specified address in an SD card, in DMA mode.

Parameters:
   pData,: Pointer to the buffer that will contain the data to transmit
   ReadAddr,: Address from where data is to be read
   NumOfBlocks,: Number of SD blocks to read

Return values:
**SD status**

Definition at line 297 of file `stm32f769i_discovery_sd.c`.

References **MSD_ERROR**, **MSD_OK**, and **uSdHandle**.

```c
void BSP_SD_ReadCpltCallback ( void )
```

BSP Rx Transfer completed callbacks.

**Return values:**

*None*

Definition at line 583 of file `stm32f769i_discovery_sd.c`.

Referenced by **HAL_SD_RxCpltCallback()**.

```c
uint8_t BSP_SD_WriteBlocks ( uint32_t * pData, uint32_t WriteAddr, uint32_t NumOfBlocks, uint32_t Timeout )
```

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

**Parameters:**

- **pData:** Pointer to the buffer that will contain the data to transmit
- **WriteAddr:** Address from where data is to be written
- **NumOfBlocks:** Number of SD blocks to write
- **Timeout:** Timeout for write operation

**Return values:**

*SD status*
Definition at line 278 of file stm32f769i_discovery_sd.c.
References MSD_ERROR, MSD_OK, and uSdHandle.

uint8_t BSP_SD_WriteBlocks_DMA ( uint32_t * pData, 
                                uint32_t WriteAddr, 
                                uint32_t NumOfBlocks 
                             )

Writes block(s) to a specified address in an SD card, in DMA mode.

Parameters:
  pData: Pointer to the buffer that will contain the data to transmit
  WriteAddr: Address from where data is to be written
  NumOfBlocks: Number of SD blocks to write

Return values:
  SD status

Definition at line 317 of file stm32f769i_discovery_sd.c.
References MSD_ERROR, MSD_OK, and uSdHandle.

void BSP_SD_WriteCpltCallback ( void )

BSP Tx Transfer completed callbacks.

Return values:
  None

Definition at line 574 of file stm32f769i_discovery_sd.c.

Referenced by HAL_SD_TxCpltCallback().
### SD Exported Types

| STM32F769I_DISCOVERY_SD | Defines |
Defines

#define BSP_SD_CardInfo HAL_SD_CardInfoTypeDef
SD Card information structure.
Define Documentation

```c
#define BSP_SD_CardInfo   HAL_SD_CardInfoTypeDef
```

SD Card information structure.

Definition at line 69 of file `stm32f769i_discovery_sd.h`.
## SDRAM Exported Functions

STM32F769I_DISCOVERY SDRAM
# Functions

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td><strong>BSP_SDRAM_Init</strong> (void)</td>
<td>Initializes the SDRAM device.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_SDRAM_DeInit</strong> (void)</td>
<td>DeInitializes the SDRAM device.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_SDRAM_Initialization_sequence</strong> (uint32_t RefreshCount)</td>
<td>Programs the SDRAM device.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_SDRAM_ReadData</strong> (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)</td>
<td>Reads an amount of data from the SDRAM memory in polling mode.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_SDRAM_ReadData_DMA</strong> (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)</td>
<td>Reads an amount of data from the SDRAM memory in DMA mode.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_SDRAM_WriteData</strong> (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)</td>
<td>Writes an amount of data to the SDRAM memory in polling mode.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_SDRAM_WriteData_DMA</strong> (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)</td>
<td>Writes an amount of data to the SDRAM memory in DMA mode.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_SDRAM_Sendcmd</strong> (FMC_SDRAM_CommandTypeDef *SdramCmd)</td>
<td>Sends command to the SDRAM bank.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_SDRAM_MspInit</strong> (SDRAM_HandleTypeDef *hsdram, void *Params)</td>
<td>Initializes SDRAM MSP.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_SDRAM_MspDeInit</strong> (SDRAM_HandleTypeDef *hsdram, void *Params)</td>
<td>DeInitializes SDRAM MSP.</td>
</tr>
</tbody>
</table>
Function Documentation

```
uint8_t BSP_SDRAM_DeInit ( void )
```

DeInitializes the SDRAM device.

**Return values:**
- `SDRAM` status

Definition at line 187 of file `stm32f769i_discovery_sdram.c`.

References `BSP_SDRAM_MspDeInit()`, `SDRAM_ERROR`, `SDRAM_OK`, and `sdramHandle`.

```
uint8_t BSP_SDRAM_Init ( void )
```

Initializes the SDRAM device.

**Return values:**
- `SDRAM` status

Definition at line 138 of file `stm32f769i_discovery_sdram.c`.

References `BSP_SDRAM_Initialization_sequence()`, `BSP_SDRAM_MspInit()`, `REFRESH_COUNT`, `SDCLOCK_PERIOD`, `SDRAM_ERROR`, `SDRAM_MEMORY_WIDTH`, `SDRAM_OK`, `sdramHandle`, and `Timing`.

Referenced by `BSP_LCD_InitEx()`.

```
void BSP_SDRAM_Initialization_sequence ( uint32_t RefreshCount )
```

Programs the SDRAM device.
Parameters:

**RefreshCount:** SDRAM refresh counter value

Return values:

None

Definition at line 213 of file `stm32f769i_discovery_sdram.c`.

References `Command`, `SDRAM_MODEREG_BURST_LENGTH_1`, `SDRAM_MODEREG_BURST_TYPESEQUENTIAL`, `SDRAM_MODEREG_CAS_LATENCY_3`, `SDRAM_MODEREG_OPERATING_MODE_STANDARD`, `SDRAM_MODEREG_WRITEBURST_MODE_SINGLE`, `SDRAM_TIMEOUT`, and `sdramHandle`.

Referenced by `BSP_SDRAM_Init()`.

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

Deinitializes SDRAM MSP.

Parameters:

**hsdram:** SDRAM handle  
**Params**

Return values:

None

Definition at line 465 of file `stm32f769i_discovery_sdram.c`.

References `SDRAM_DMAx_IRQn`, and `SDRAM_DMAx_STREAM`.

Referenced by `BSP_SDRAM_DeInit()`.
void **BSP_SDRAM_MspInit** (SDRAM_HandleTypeDef * hsdram,
void * Params
)

Initializes SDRAM MSP.

**Parameters:**
- **hsdram:** SDRAM handle
- **Params**

**Return values:**
- **None**

Definition at line 367 of file **stm32f769i_discovery_sdram.c**.

References **__DMAx_CLK_ENABLE**, **SDRAM_DMAx_CHANNEL**, **SDRAM_DMAx_IRQn**, and **SDRAM_DMAx_STREAM**.

Referenced by **BSP_SDRAM_Init**().

```c
uint8_t **BSP_SDRAM_ReadData** (uint32_t uwStartAddress,
uint32_t * pData,
uint32_t uwDataSize
)
```

Reads an amount of data from the SDRAM memory in polling mode.

**Parameters:**
- **uwStartAddress:** Read start address
- **pData:** Pointer to data to be read
- **uwDataSize:** Size of read data from the memory

**Return values:**
- **SDRAM** status
uint8_t BSP_SDRAM_ReadData_DMA ( uint32_t uwStartAddress, 
                                  uint32_t * pData, 
                                  uint32_t uwDataSize 
                              )

Reads an amount of data from the SDRAM memory in DMA mode.

**Parameters:**
- **uwStartAddress:** Read start address
- **pData:** Pointer to data to be read
- **uwDataSize:** Size of read data from the memory

**Return values:**
- **SDRAM** status

Definition at line 275 of file stm32f769i_discovery_sdram.c.

References **SDRAM_ERROR**, **SDRAM_OK**, and **sdramHandle**.

uint8_t BSP_SDRAM_Sendcmd ( FMC_SDRAM_CommandTypeDef * SdramCmd, 
                           
                           )

Sends command to the SDRAM bank.

**Parameters:**
- **SdramCmd:** Pointer to SDRAM command structure

**Return values:**
- **SDRAM** status

Definition at line 294 of file stm32f769i_discovery_sdram.c.

References **SDRAM_ERROR**, **SDRAM_OK**, and **sdramHandle**.

uint8_t BSP_SDRAM_Sendcmd ( FMC_SDRAM_CommandTypeDef * SdramCmd, 
                           
                           )

Sends command to the SDRAM bank.

**Parameters:**
- **SdramCmd:** Pointer to SDRAM command structure

**Return values:**
- **SDRAM** status

Definition at line 349 of file stm32f769i_discovery_sdram.c.

References **SDRAM_ERROR**, **SDRAM_OK**, and **sdramHandle**.
uint8_t BSP_SDRAM_WriteData ( uint32_t uwStartAddress, 
   uint32_t * pData, 
   uint32_t uwDataSize 
)

Writes an amount of data to the SDRAM memory in polling mode.

**Parameters:**
- **uwStartAddress:** Write start address
- **pData:** Pointer to data to be written
- **uwDataSize:** Size of written data from the memory

**Return values:**
- **SDRAM** status

Definition at line 313 of file stm32f769i_discovery_sdram.c.

References **SDRAM_ERROR, SDRAM_OK, and sdramHandle.**

uint8_t BSP_SDRAM_WriteData_DMA ( uint32_t uwStartAddress, 
   uint32_t * pData, 
   uint32_t uwDataSize 
)

Writes an amount of data to the SDRAM memory in DMA mode.

**Parameters:**
- **uwStartAddress:** Write start address
- **pData:** Pointer to data to be written
- **uwDataSize:** Size of written data from the memory
Return values:

**SDRAM** status

Definition at line **332** of file **stm32f769i_discovery_sdram.c**.

References **SDRAM_ERROR**, **SDRAM_OK**, and **sdramHandle**.
## SDRAM Private Functions

STM32F769I_DISCOVERY SDRAM
## Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uint8_t</code> BSP_SDRAM_Init (void)</td>
<td>Initializes the SDRAM device.</td>
</tr>
<tr>
<td><code>uint8_t</code> BSP_SDRAM_DeInit (void)</td>
<td>DeInitializes the SDRAM device.</td>
</tr>
<tr>
<td><code>void</code> BSP_SDRAM_Initialization_sequence (uint32_t RefreshCount)</td>
<td>Programs the SDRAM device.</td>
</tr>
<tr>
<td><code>uint8_t</code> BSP_SDRAM_ReadData (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)</td>
<td>Reads an amount of data from the SDRAM memory in polling mode.</td>
</tr>
<tr>
<td><code>uint8_t</code> BSP_SDRAM_ReadData_DMA (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)</td>
<td>Reads an amount of data from the SDRAM memory in DMA mode.</td>
</tr>
<tr>
<td><code>uint8_t</code> BSP_SDRAM_WriteData (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)</td>
<td>Writes an amount of data to the SDRAM memory in polling mode.</td>
</tr>
<tr>
<td><code>uint8_t</code> BSP_SDRAM_WriteData_DMA (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)</td>
<td>Writes an amount of data to the SDRAM memory in DMA mode.</td>
</tr>
<tr>
<td><code>uint8_t</code> BSP_SDRAM_Sendcmd (FMC_SDRAM_CommandTypeDef *SdramCmd)</td>
<td>Sends command to the SDRAM bank.</td>
</tr>
<tr>
<td><code>__weak void</code> BSP_SDRAM_MspInit (SDRAM_HandleTypeDef *hsdram, void *Params)</td>
<td>Initializes SDRAM MSP.</td>
</tr>
<tr>
<td><code>__weak void</code> BSP_SDRAM_MspDeInit (SDRAM_HandleTypeDef *hsdram, void *Params)</td>
<td>DeInitializes SDRAM MSP.</td>
</tr>
</tbody>
</table>
Function Documentation

\textbf{uint8_t BSP_SDRAM_DeInit ( void )}

DeInitializes the SDRAM device.

\textbf{Return values:}

\textbf{SDRAM} status

Definition at line 187 of file \textit{stm32f769i_discovery_sdram.c}.

References \textbf{BSP_SDRAM_MspDeInit()}, \textbf{SDRAM_ERROR}, \textbf{SDRAM_OK}, and \textit{sdramHandle}.

\textbf{uint8_t BSP_SDRAM_Init ( void )}

Initializes the SDRAM device.

\textbf{Return values:}

\textbf{SDRAM} status

Definition at line 138 of file \textit{stm32f769i_discovery_sdram.c}.


Referenced by \textbf{BSP_LCD_InitEx()}. 

\textbf{void BSP_SDRAM_Initialization_sequence ( uint32_t RefreshCount )}

Programs the SDRAM device.
Parameters:

**RefreshCount**: SDRAM refresh counter value

Return values:

None

Definition at line 213 of file `stm32f769i_discovery_sdram.c`.


Referenced by **BSP_SDRAM_Init()**.

```c
__weak void BSP_SDRAM_MspDeInit ( SDRAM_HandleTypeDef * hsdram, void * Params )
```

DeInitializes SDRAM MSP.

Parameters:

**hsdram**: SDRAM handle
**Params**

Return values:

None

Definition at line 465 of file `stm32f769i_discovery_sdram.c`.

References **SDRAM_DMAx_IRQn**, and **SDRAM_DMAx_STREAM**.

Referenced by **BSP_SDRAM_DeInit()**.
__weak void BSP_SDRAM_MspInit ( SDRAM_HandleTypeDef * hsdram,
                                void * Params)

Initializes SDRAM MSP.

Parameters:
  hsdram,:	SDRAM handle
  Params

Return values:
  None

Definition at line 367 of file stm32f769i_discovery_sdram.c.

References __DMAx_CLK_ENABLE, SDRAM_DMAx_CHANNEL, SDRAM_DMAx_IRQn, and SDRAM_DMAx_STREAM.

Referenced by BSP_SDRAM_Init().

uint8_t BSP_SDRAM_ReadData ( uint32_t uwStartAddress,  
                            uint32_t * pData,  
                            uint32_t uwDataSize)

Reads an amount of data from the SDRAM memory in polling mode.

Parameters:
  uwStartAddress,: Read start address
  pData,: Pointer to data to be read
  uwDataSize,: Size of read data from the memory

Return values:
  SDRAM status
uint8_t BSP_SDRAM_ReadData_DMA (uint32_t uwStartAddress, uint32_t * pData, uint32_t uwDataSize)

Reads an amount of data from the SDRAM memory in DMA mode.

Parameters:
- `uwStartAddress`: Read start address
- `pData`: Pointer to data to be read
- `uwDataSize`: Size of read data from the memory

Return values:
- `SDRAM` status

uint8_t BSP_SDRAM_Sendcmd (FMC_SDRAM_CommandTypeDef * SdramCmd)

Sends command to the SDRAM bank.

Parameters:
- `SdramCmd`: Pointer to SDRAM command structure

Return values:
- `SDRAM` status

Definition at line 275 of file `stm32f769i_discovery_sdram.c`.
References `SDRAM_ERROR`, `SDRAM_OK`, and `sdramHandle`.

Definition at line 294 of file `stm32f769i_discovery_sdram.c`.
References `SDRAM_ERROR`, `SDRAM_OK`, and `sdramHandle`.

Definition at line 349 of file `stm32f769i_discovery_sdram.c`.
References **SDRAM_ERROR, SDRAM_OK, SDRAM_TIMEOUT**, and **sdramHandle**.

```c
uint8_t BSP_SDRAM_WriteData ( uint32_t uwStartAddress, uint32_t * pData, uint32_t uwDataSize )
```

Writes an amount of data to the SDRAM memory in polling mode.

**Parameters:**
- **uwStartAddress**: Write start address
- **pData**: Pointer to data to be written
- **uwDataSize**: Size of written data from the memory

**Return values:**
- SDRAM status

Definition at line 313 of file **stm32f769i_discovery_sdram.c**.

References **SDRAM_ERROR, SDRAM_OK**, and **sdramHandle**.

```c
uint8_t BSP_SDRAM_WriteData_DMA ( uint32_t uwStartAddress, uint32_t * pData, uint32_t uwDataSize )
```

Writes an amount of data to the SDRAM memory in DMA mode.

**Parameters:**
- **uwStartAddress**: Write start address
- **pData**: Pointer to data to be written
- **uwDataSize**: Size of written data from the memory
Return values:

**SDRAM** status

Definition at line 332 of file `stm32f769i_discovery_sdram.c`.

References **SDRAM_ERROR**, **SDRAM_OK**, and `sdramHandle`. 
# STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td>Functions</td>
</tr>
</tbody>
</table>

## TS Exported Functions

STM32F769I_DISCOVERY TS
## Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_TS_Init</strong> (uint16_t ts_SizeX, uint16_t ts_SizeY)</td>
<td>Initializes and configures the touch screen functionalities and configures all necessary hardware resources (GPIOs, I2C, clocks..).</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_TS_GetState</strong> (TS_StateTypeDef *TS_State)</td>
<td>Returns status and positions of the touch screen.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><strong>BSP_TS_ITConfig</strong> (void)</td>
<td>Configures and enables the touch screen interrupts.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><strong>BSP_TS_INT_MspInit</strong> (void)</td>
<td>Initializes the TS_INT pin MSP.</td>
</tr>
</tbody>
</table>
Function Documentation

```c
uint8_t BSP_TS_GetState ( TS_StateTypeDef * TS_State )
```

Returns status and positions of the touch screen.

**Parameters:**
- `TS_State`: Pointer to touch screen current state structure

**Return values:**
- `TS_OK` if all initializations are OK. Other value if error.

Definition at line 257 of file `stm32f769i_discovery_ts.c`.

References `I2C_Address`, `TOUCH_EVENT_CONTACT`, `TOUCH_EVENT_LIFT_UP`, `TOUCH_EVENT_NO_EVT`, `TOUCH_EVENT_PRESS_DOWN`, `TS_StateTypeDef::touchDetected`, `TS_StateTypeDef::touchX`, `TS_StateTypeDef::touchY`, `ts_driver`, `TS_ERROR`, `TS_MAX_NB_TOUCH`, `TS_OK`, `ts_orientation`, `TS_SWAP_X`, `TS_SWAP_XY`, and `TS_SWAP_Y`.

```c
uint8_t BSP_TS_Init ( uint16_t ts_SizeX, uint16_t ts_SizeY )
```

Initializes and configures the touch screen functionalities and configures all necessary hardware resources (GPIOs, I2C, clocks..).

**Parameters:**
- `ts_SizeX` : Maximum X size of the TS area on LCD
- `ts_SizeY` : Maximum Y size of the TS area on LCD

**Return values:**
**TS_OK** if all initializations are OK. Other value if error.

Definition at line 162 of file *stm32f769i_discovery_ts.c*.


```c
void BSP_TS_INT_MspInit ( void )
```

Initializes the TS_INT pin MSP.

**Return values:**

None

Definition at line 446 of file *stm32f769i_discovery_ts.c*.

References **TS_INT_GPIO_CLK_ENABLE**, **TS_INT_GPIO_PORT**, and **TS_INT_PIN**.

Referenced by **BSP_TS_ITConfig()**.

```c
uint8_t BSP_TS_ITConfig ( void )
```

Configures and enables the touch screen interrupts.

**Return values:**

**TS_OK** if all initializations are OK. Other value if error.

Definition at line 223 of file *stm32f769i_discovery_ts.c*.

References **BSP_TS_INT_MspInit()**, **I2C_Address**, **ts_driver**, **TS_INT_EXTI_IRQn**, **TS_INT_GPIO_PORT**, **TS_INT_PIN**, and **TS_OK**.
# STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## TS Public Functions

STM32F769I_DISCOVERY TS
## Modules

| TS Private Functions |
### Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td><strong>BSP_TS_Init</strong> (uint16_t ts_SizeX, uint16_t ts_SizeY)</td>
<td>Initializes and configures the touch screen functionalities and configures all necessary hardware resources (GPIOs, I2C, clocks..).</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_TS_ITConfig</strong> (void)</td>
<td>Configures and enables the touch screen interrupts.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_TS_GetState</strong> (TS_StateTypeDef *TS_State)</td>
<td>Returns status and positions of the touch screen.</td>
</tr>
</tbody>
</table>
Function Documentation

```c
uint8_t BSP_TS_GetState ( TS_StateTypeDef * TS_State )
```

Returns status and positions of the touch screen.

**Parameters:**

- `TS_State`: Pointer to touch screen current state structure

**Return values:**

- `TS_OK` if all initializations are OK. Other value if error.

Definition at line 257 of file `stm32f769i_discovery_ts.c`.

References `I2C_Address`, `TOUCH_EVENT_CONTACT`, `TOUCH_EVENT_LIFT_UP`, `TOUCH_EVENT_NO_EVT`, `TOUCH_EVENT_PRESS_DOWN`, `TS_StateTypeDef::touchDetected`, `TS_StateTypeDef::touchX`, `TS_StateTypeDef::touchY`, `ts_driver`, `TS_ERROR`, `TS_MAX_NB_TOUCH`, `TS_OK`, `ts_orientation`, `TS_SWAP_X`, `TS_SWAP_XY`, and `TS_SWAP_Y`.

```c
uint8_t BSP_TS_Init ( uint16_t ts_SizeX, 
                    uint16_t ts_SizeY )
```

Initializes and configures the touch screen functionalities and configures all necessary hardware resources (GPIOs, I2C, clocks..).

**Parameters:**

- `ts_SizeX` : Maximum X size of the TS area on LCD
- `ts_SizeY` : Maximum Y size of the TS area on LCD

**Return values:**
**TS_OK** if all initializations are OK. Other value if error.

Definition at line 162 of file `stm32f769i_discovery_ts.c`.


```c
uint8_t BSP_TS_ITConfig ( void )
```

Configures and enables the touch screen interrupts.

**Return values:**

**TS_OK** if all initializations are OK. Other value if error.

Definition at line 223 of file `stm32f769i_discovery_ts.c`.

References **BSP_TS_INT_MspInit()**, **I2C_Address**, **ts_driver**, **TS_INT_EXTI_IRQn**, **TS_INT_GPIO_PORT**, **TS_INT_PIN**, and **TS_OK**.
# STM32F769I-Discovery BSP User Manual

## Directories

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
</table>

**TS Private Functions**

**TS Public Functions**
Functions

__weak void BSP_TS_INT_MspInit (void)
Initializes the TS_INT pin MSP.
Function Documentation

__weak void BSP_TS_INT_MspInit ( void )

Initializes the TS_INT pin MSP.

Return values:
   None

Definition at line 446 of file stm32f769i_discovery_ts.c.

References TS_INT_GPIO_CLK_ENABLE, TS_INT_GPIO_PORT, and TS_INT_PIN.

Referenced by BSP_TS_ITConfig().
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

STM32F769I Discovery Low Level Button

STM32F769I Discovery Low Level Exported Constants

Defines
#define BUTTONn ((uint8_t)1)
#define WAKEUP_BUTTON_PIN GPIO_PIN_0
Wakeup push-button.

#define WAKEUP_BUTTON_GPIO_PORT GPIOA
#define WAKEUP_BUTTON_GPIO_CLK_ENABLE() __HAL_RCC_GPIOA_CLK_ENABLE()
#define WAKEUP_BUTTON_GPIO_CLK_DISABLE() __HAL_RCC_GPIOA_CLK_DISABLE()
#define WAKEUP_BUTTON_EXTI_IRQn EXTI0_IRQn
#define USER_BUTTON_PIN WAKEUP_BUTTON_PIN
#define USER_BUTTON_GPIO_PORT WAKEUP_BUTTON_GPIO_PORT
#define USER_BUTTON_GPIO_CLK_ENABLE() WAKEUP_BUTTON_GPIO_CLK_ENABLE()
#define USER_BUTTON_GPIO_CLK_DISABLE() WAKEUP_BUTTON_GPIO_CLK_DISABLE()
#define USER_BUTTON_EXTI_IRQn WAKEUP_BUTTON_EXTI_IRQn
#define BUTTON_GPIO_CLK_ENABLE() __HAL_RCC_GPIOA_CLK_ENABLE()
Define Documentation

```
#define BUTTON_GPIO_CLK_ENABLE (__HAL_RCC_GPIOA_CLK_ENABLE())
Definition at line 177 of file stm32f769i_discovery.h.
Referenced by BSP_PB_Init().
```

```
#define BUTTONn ((uint8_t)1)
Definition at line 159 of file stm32f769i_discovery.h.
```

```
#define USER_BUTTON EXTI IRQn  WAKEUP_BUTTON EXTI IRQn
Definition at line 175 of file stm32f769i_discovery.h.
```

```
#define USER_BUTTON_GPIO_CLK_DISABLE ( )  WAKEUP_BUTTON_GPIO_CLK_DISABLE
Definition at line 174 of file stm32f769i_discovery.h.
```

```
#define USER_BUTTON_GPIO_CLK_ENABLE ( )  WAKEUP_BUTTON_GPIO_CLK_ENABLE
Definition at line 173 of file stm32f769i_discovery.h.
```

```
#define USER_BUTTON_GPIO_PORT  WAKEUP_BUTTON_GPIO_PORT
Definition at line 172 of file stm32f769i_discovery.h.
```

```
#define USER_BUTTON_PIN  WAKEUP_BUTTON_PIN
```
Definition at line 171 of file `stm32f769i_discovery.h`.

```c
#define WAKEUP_BUTTON_EXTI_IRQHandler EXTI0_IRQHandler
```

Definition at line 168 of file `stm32f769i_discovery.h`.

```c
#define WAKEUP_BUTTON_GPIO_CLK_DISABLE () __HAL_RCC_GPIOA_CLK_DISABLE()
```

Definition at line 167 of file `stm32f769i_discovery.h`.

```c
#define WAKEUP_BUTTON_GPIO_CLK_ENABLE () __HAL_RCC_GPIOA_CLK_ENABLE()
```

Definition at line 166 of file `stm32f769i_discovery.h`.

```c
#define WAKEUP_BUTTON_GPIO_PORT GPIOA
```

Definition at line 165 of file `stm32f769i_discovery.h`.

```c
#define WAKEUP_BUTTON_PIN GPIO_PIN_0
```

Wakeup push-button.

Definition at line 164 of file `stm32f769i_discovery.h`.

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
## LOW_LEVEL Private Variables

**STM32F769I_DISCOVERY LOW LEVEL**
## Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uint32_t</code></td>
<td><code>GPIO_PIN [LEDn]</code></td>
</tr>
<tr>
<td><code>GPIO_TypeDef *</code></td>
<td><code>GPIO_PORT [LEDn]</code></td>
</tr>
<tr>
<td><code>GPIO_TypeDef *</code></td>
<td><code>BUTTON_PORT [BUTTONn] = {WAKEUP_BUTTON_GPIO_PORT}</code></td>
</tr>
<tr>
<td><code>const uint16_t</code></td>
<td><code>BUTTON_PIN [BUTTONn] = {WAKEUP_BUTTON_PIN}</code></td>
</tr>
<tr>
<td><code>const uint16_t</code></td>
<td><code>BUTTON_IRQHandler [BUTTONn] = {WAKEUP_BUTTON_EXTI_IRQHandler}</code></td>
</tr>
<tr>
<td><code>static I2C_HandleTypeDef</code></td>
<td><code>hI2cAudioHandler = {0}</code></td>
</tr>
<tr>
<td><code>static I2C_HandleTypeDef</code></td>
<td><code>hI2cExtHandler = {0}</code></td>
</tr>
</tbody>
</table>
Variable Documentation

**const uint16_t** `BUTTON_IRQn[BUTTONn] = {WAKEUP_BUTTON_E}`

Definition at line 101 of file `stm32f769i_discovery.c`.

Referenced by `BSP_PB_DeInit()`, and `BSP_PB_Init()`.

**const uint16_t** `BUTTON_PIN[BUTTONn] = {WAKEUP_BUTTON_PIN`}

Definition at line 99 of file `stm32f769i_discovery.c`.

Referenced by `BSP_PB_DeInit()`, `BSP_PB_GetState()`, and `BSP_PB_Init()`.

`GPIO_TypeDef*` `BUTTON_PORT[BUTTONn] = {WAKEUP_BUTTON_`

Definition at line 97 of file `stm32f769i_discovery.c`.

Referenced by `BSP_PB_DeInit()`, `BSP_PB_GetState()`, and `BSP_PB_Init()`.

**uint32_t** `GPIO_PIN[LEDn]`

Initial value:

```
{LED1_PIN,
 LED2_PIN}
```

Definition at line 91 of file `stm32f769i_discovery.c`.

Referenced by `BSP_LED_DeInit()`, `BSP_LED_Init()`, `BSP_LED_Off()`, `BSP_LED_On()`, and `BSP_LED_Toggle()`.
GPIO_TypeDef* GPIO_PORT[LEDn]

Initial value:

```
{LED1_GPIO_PORT,
 LED2_GPIO_PORT}
```

Definition at line 94 of file stm32f769i_discovery.c.

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

I2C_HandleTypeDef hI2cAudioHandler = {0} [static]

Definition at line 104 of file stm32f769i_discovery.c.

Referenced by AUDIO_IO_Init(), AUDIO_IO_Read(),
AUDIO_IO_Write(), HDMI_IO_Init(), HDMI_IO_Read(),
HDMI_IO_Write(), I2Cx_Init(), I2Cx_MspInit(), TS_IO_Init(),
TS_IO_Read(), TS_IO_ReadMultiple(), TS_IO_Write(), and
TS_IO_WriteMultiple().

I2C_HandleTypeDef hI2cExtHandler = {0} [static]

Definition at line 105 of file stm32f769i_discovery.c.

Referenced by EEPROM_IO_Init(), EEPROM_IO_IsDeviceReady(),
EEPROM_IO_ReadData(), and EEPROM_IO_WriteData().
# STM32F769I-Discovery BSP User Manual

## Exported_types

STM32F769I Discovery Low Level Exported Types
## Enumerations

<table>
<thead>
<tr>
<th>Enum</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ButtonValue_TypeDef</td>
<td><code>enum ButtonValue_TypeDef { PB_SET = 0, PB_RESET = !PB_SET }</code></td>
</tr>
<tr>
<td>DISCO_Status_TypeDef</td>
<td><code>enum DISCO_Status_TypeDef { DISCO_OK = 0, DISCO_ERROR = 1 }</code></td>
</tr>
</tbody>
</table>

DISCO_Status_TypeDef STM32F769I_DISCO board Status return possible values. More...
Enumeration Type Documentation

enum ButtonValue_TypeDef

**Enumerator:**

PB_SET
PB_RESET

Definition at line 108 of file stm32f769i_discovery.h.

enum DISCO_Status_TypeDef

DISCO_Status_TypeDef STM32F769I_DISCO board Status return possible values.

**Enumerator:**

DISCO_OK
DISCO_ERROR

Definition at line 118 of file stm32f769i_discovery.h.

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery
BSP User Manual by doxygen 1.7.6.1
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SDRAM Private Variables</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

STM32F769I_DISCOVERY SDRAM
### Variables

<table>
<thead>
<tr>
<th>TypeDefinition</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDRAM_HandleTypeDef</td>
<td>sdramHandle</td>
</tr>
<tr>
<td>static FMC_SDRAM_TimingTypeDef</td>
<td>Timing</td>
</tr>
<tr>
<td>static FMC_SDRAM_CommandTypeDef</td>
<td>Command</td>
</tr>
</tbody>
</table>
Variable Documentation

**FMC_SDRAM_CommandTypeDef Command** [static]

Definition at line 118 of file stm32f769i_discovery_sdram.c.

Referenced by **BSP_SDRAM_Initialization_sequence()**.

**SDRAM_HandleTypeDef sramHandle**

Definition at line 116 of file stm32f769i_discovery_sdram.c.

Referenced by **BSP_SDRAM_DeInit()**, **BSP_SDRAM_Init()**, **BSP_SDRAM_Initialization_sequence()**, **BSP_SDRAM_ReadData()**, **BSP_SDRAM_ReadData_DMA()**, **BSP_SDRAM_Sendcmd()**, **BSP_SDRAM_WriteData()**, and **BSP_SDRAM_WriteData_DMA()**.

**FMC_SDRAM_TimingTypeDef Timing** [static]

Definition at line 117 of file stm32f769i_discovery_sdram.c.

Referenced by **BSP_SDRAM_Init()**.
STM32F769I-Discovery BSP User Manual

STM32F769I_DISCOVERY_AUDIO Private Macros

STM32F769I_DISCOVERY_AUDIO
Defines

```c
#define DFSDM_OVER_Sampling(__FREQUENCY__)
#define DFSDM_CLOCK_DIVIDER(__FREQUENCY__)  
#define DFSDM_FILTER_ORDER(__FREQUENCY__)   
#define DFSDM_RIGHT_BIT_SHIFT(__FREQUENCY__) 
#define SaturaLH(N, L, H) ((((N)<(L))?(L):(((N)>(H))?(H):(N))));
```
Define Documentation

```c
#define DFSDM_CLOCK_DIVIDER(__FREQUENCY__)
Value:

(__FREQUENCY__ == AUDIO_FREQUENCY_8K) ? 24 :
   (__FREQUENCY__ == AUDIO_FREQUENCY_11K) ? 4 :
   (__FREQUENCY__ == AUDIO_FREQUENCY_16K) ? 24 :
   (__FREQUENCY__ == AUDIO_FREQUENCY_22K) ? 4 :
   (__FREQUENCY__ == AUDIO_FREQUENCY_32K) ? 24 :
   (__FREQUENCY__ == AUDIO_FREQUENCY_44K) ? 4 :
   (__FREQUENCY__ == AUDIO_FREQUENCY_48K) ? 25 :
   25

Definition at line 179 of file stm32f769i_discovery_audio.c.
Referenced by DFSDMx_Init().
```

```c
#define DFSDM_FILTER_ORDER(__FREQUENCY__)
Value:

(__FREQUENCY__ == AUDIO_FREQUENCY_8K) ? DFSDM_FILTER_SINC3_ORDER :
   (__FREQUENCY__ == AUDIO_FREQUENCY_11K) ? DFSDM_FILTER_SINC3_ORDER :
   (__FREQUENCY__ == AUDIO_FREQUENCY_16K) ? DFSDM_FILTER_SINC3_ORDER :
   (__FREQUENCY__ == AUDIO_FREQUENCY_22K) ? DFSDM_FILTER_SINC3_ORDER :
   (__FREQUENCY__ == AUDIO_FREQUENCY_32K) ? DFSDM_FILTER_SINC4_ORDER :
   DFSDM_FILTER_SINC3_ORDER
```

DFSDM_FILTER_SINC3_ORDER

: \(_{\text{FREQUENCY}} \) == \(_{\text{AUDIO\_FREQUENCY\_44}\text{K}}\) ?

: \(_{\text{FREQUENCY}} \) == \(_{\text{AUDIO\_FREQUENCY\_48}\text{K}}\) ?

DFSDM_FILTER_SINC3_ORDER : DFSDM_FILTER_SINC5_ORDER

Definition at line 188 of file \text{stm32f769i\_discovery\_audio.c}.

Referenced by \text{DFSDMx\_Init()}. 

\#define CFREQ\_OVER\_SAMPLING( \(_{\text{FREQUENCY}}\))

\text{Value:}

\(_{\text{FREQUENCY}} \) == \(_{\text{AUDIO\_FREQUENCY\_8}\text{K}}\) ? 256 \ \

256 \ \

: \(_{\text{FREQUENCY}} \) == \(_{\text{AUDIO\_FREQUENCY\_11}\text{K}}\) ?

128 \ \

: \(_{\text{FREQUENCY}} \) == \(_{\text{AUDIO\_FREQUENCY\_16}\text{K}}\) ?

128 \ \

: \(_{\text{FREQUENCY}} \) == \(_{\text{AUDIO\_FREQUENCY\_22}\text{K}}\) ?

64 \ \

: \(_{\text{FREQUENCY}} \) == \(_{\text{AUDIO\_FREQUENCY\_32}\text{K}}\) ?

64 \ \

: \(_{\text{FREQUENCY}} \) == \(_{\text{AUDIO\_FREQUENCY\_44}\text{K}}\) ?

40 \ : 20 \ \\

Definition at line 170 of file \text{stm32f769i\_discovery\_audio.c}.

Referenced by \text{DFSDMx\_Init()}. 

\#define CFREQ\_RIGHT\_BIT\_SHIFT( \(_{\text{FREQUENCY}}\))

\text{Value:}

\(_{\text{FREQUENCY}} \) == \(_{\text{AUDIO\_FREQUENCY\_8}\text{K}}\) ? 8 \ \

: \(_{\text{FREQUENCY}} \) == \(_{\text{AUDIO\_FREQUENCY\_11}\text{K}}\) ?


```
8 \n  : (__FREQUENCY__ == AUDIO_FREQUENCY_16K) ?
3 \n  : (__FREQUENCY__ == AUDIO_FREQUENCY_22K) ?
4 \n  : (__FREQUENCY__ == AUDIO_FREQUENCY_32K) ?
7 \n  : (__FREQUENCY__ == AUDIO_FREQUENCY_44K) ?
0 \n  : (__FREQUENCY__ == AUDIO_FREQUENCY_48K) ?
0 : 4 \
```

Definition at line 197 of file stm32f769i_discovery_audio.c.

Referenced by DFSDMx_Init().

```c
#define SaturaLH ( N, 
    L, 
    H 
  ) (((N)<(L))?(L):(((N)>(H))?(H):(N)))
```

Definition at line 207 of file stm32f769i_discovery_audio.c.

Referenced by HAL_DFSDM_FilterRegConvCpltCallback(), and HAL_DFSDM_FilterRegConvHalfCpltCallback().

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery
BSP User Manual by doxygen 1.7.6.1
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**STM32F769I_DISCOVERY_AUDIO Exported Macros**

STM32F769I_DISCOVERY_AUDIO

Defines
#define DMA_MAX(x)  (((x) <= DMA_MAX_SZE)? (x):DMA_MAX_SZE)
Define Documentation

#define DMA_MAX ( x ) (((x) <= DMA_MAX_SZE)? (x):DMA_MAX_SZE)

Definition at line 250 of file stm32f769i_discovery_audio.h.

Referenced by BSP_AUDIO_OUT_Play().

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery
BSP User Manual by doxygen 1.7.6.1
# STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## EEPROM Exported Constants

**STM32F769I_DISCOVERY EEPROM**
Defines

<table>
<thead>
<tr>
<th>Defines</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>EEPROM_PAGESIZE</code></td>
<td><code>(uint8_t)4</code></td>
</tr>
<tr>
<td><code>EEPROM_MAX_SIZE</code></td>
<td><code>(uint16_t)0x2000 /* 64Kbit */</code></td>
</tr>
<tr>
<td><code>EEPROM_MAX_TRIALS</code></td>
<td><code>(uint32_t)3000</code></td>
</tr>
<tr>
<td><code>EEPROM_OK</code></td>
<td><code>(uint32_t)0</code></td>
</tr>
<tr>
<td><code>EEPROM_FAIL</code></td>
<td><code>(uint32_t)1</code></td>
</tr>
<tr>
<td><code>EEPROM_TIMEOUT</code></td>
<td><code>(uint32_t)2</code></td>
</tr>
</tbody>
</table>
Define Documentation

```c
#define EEPROM_FAIL  ((uint32_t)1)
```
Definition at line 82 of file stm32f769i_discovery_eeprom.h.

Referenced by `BSP_EEPROM_Init()`, `BSP_EEPROM_ReadBuffer()`, and `BSP_EEPROM_WritePage()`.

```c
#define EEPROM_MAX_SIZE  ((uint16_t)0x2000) /* 64Kbit */
```
Definition at line 75 of file stm32f769i_discovery_eeprom.h.

```c
#define EEPROM_MAX_TRIALS  ((uint32_t)3000)
```
Definition at line 79 of file stm32f769i_discovery_eeprom.h.

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

```c
#define EEPROM_OK  ((uint32_t)0)
```
Definition at line 81 of file stm32f769i_discovery_eeprom.h.

Referenced by `BSP_EEPROM_DeInit()`, `BSP_EEPROM_Init()`, `BSP_EEPROM_ReadBuffer()`, `BSP_EEPROM_WaitEepromStandbyState()`, `BSP_EEPROM_WriteBuffer()`, and `BSP_EEPROM_WritePage()`.

```c
#define EEPROM_PAGESIZE  ((uint8_t)4)
```
Definition at line 74 of file stm32f769i_discovery_eeprom.h.
Referenced by `BSP_EEPROM_WriteBuffer()`.

```
define EEPROM_TIMEOUT ((uint32_t)2)
```

Definition at line 83 of file `stm32f769i_discovery_eeprom.h`.

Referenced by `BSP_EEPROM_WaitEepromStandbyState()`.
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Directories</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEPROM Private Variables</td>
<td></td>
</tr>
</tbody>
</table>

STM32F769I_DISCOVERY EEPROM
Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>__IO uint16_t</td>
<td>EEPROMAddress</td>
<td>= 0</td>
</tr>
<tr>
<td>__IO uint16_t</td>
<td>EEPROMDataRead</td>
<td></td>
</tr>
<tr>
<td>__IO uint8_t</td>
<td>EEPROMDataWrite</td>
<td></td>
</tr>
</tbody>
</table>
Variable Documentation

__IO uint16_t EEPROMAddress = 0

Definition at line 128 of file stm32f769i_discovery_eeprom.c.

Referenced by BSP_EEPROM_Init(), BSP_EEPROM_ReadBuffer(), BSP_EEPROM_WaitEepromStandbyState(), and BSP_EEPROM_WritePage().

__IO uint16_t EEPROMDataRead

Definition at line 129 of file stm32f769i_discovery_eeprom.c.

Referenced by BSP_EEPROM_ReadBuffer().

__IO uint8_t EEPROMDataWrite

Definition at line 130 of file stm32f769i_discovery_eeprom.c.

Referenced by BSP_EEPROM_WritePage().
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**STM32F769I DISCOVERY LCD Exported Variables**

STM32F769I_DISCOVERY_LCD | STM32F769I DISCOVERY LCD Exported Macro
### Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA2D_HandleTypeDef</td>
<td>hdma2d_discovery</td>
<td></td>
</tr>
<tr>
<td>LTDC_HandleTypeDef</td>
<td>hltdc_discovery</td>
<td></td>
</tr>
<tr>
<td>DSI_HandleTypeDef</td>
<td>hdsi_discovery</td>
<td></td>
</tr>
<tr>
<td>uint32_t</td>
<td>lcd_x_size = OTM8009A_800X480_WIDTH</td>
<td></td>
</tr>
<tr>
<td>uint32_t</td>
<td>lcd_y_size = OTM8009A_800X480_HEIGHT</td>
<td></td>
</tr>
<tr>
<td>DMA2D_HandleTypeDef</td>
<td>hdma2d_discovery</td>
<td></td>
</tr>
</tbody>
</table>
Variable Documentation

**DMA2D_HandleTypeDef**  
`hdma2d_discovery`

Definition at line 178 of file `stm32f769i_discovery_lcd.c`.  
Referenced by `LL_ConvertLineToARGB8888()`, and `LL_FillBuffer()`.

**DMA2D_HandleTypeDef**  
`hdma2d_discovery`

Definition at line 178 of file `stm32f769i_discovery_lcd.c`.  
Referenced by `LL_ConvertLineToARGB8888()`, and `LL_FillBuffer()`.

**DSI_HandleTypeDef**  
`hdsi_discovery`

Definition at line 180 of file `stm32f769i_discovery_lcd.c`.  
Referenced by `BSP_LCD_DisplayOff()`, `BSP_LCD_DisplayOn()`,  
`BSP_LCD_InitEx()`, `BSP_LCD_SetBrightness()`, and  
`DSI_IO_WriteCmd()`.

**LTDC_HandleTypeDef**  
`hltdc_discovery`

Definition at line 179 of file `stm32f769i_discovery_lcd.c`.  
Referenced by `BSP_LCD_Clear()`, `BSP_LCD_DrawBitmap()`,  
`BSP_LCD_DrawHLine()`, `BSP_LCD_DrawPixel()`,  
`BSP_LCD_DrawVLine()`, `BSP_LCD_FillRect()`, `BSP_LCD_InitEx()`,  
`BSP_LCD_LayerDefaultInit()`, `BSP_LCD_ReadPixel()`,  
`BSP_LCD_ResetColorKeying()`, `BSP_LCD_SetColorKeying()`,  
`BSP_LCD_SetLayerAddress()`, `BSP_LCD_SetLayerVisible()`,  
`BSP_LCD_SetLayerWindow()`, `BSP_LCD_SetTransparency()`,
BSP_LCD_SetXSize(), and BSP_LCD_SetYSize().

```c
uint32_t lcd_x_size = OTM8009A_800X480_WIDTH
```
Definition at line 181 of file `stm32f769i_discovery_lcd.c`.
Referenced by BSP_LCD_GetXSize(), and BSP_LCD_InitEx().

```c
uint32_t lcd_y_size = OTM8009A_800X480_HEIGHT
```
Definition at line 182 of file `stm32f769i_discovery_lcd.c`.
Referenced by BSP_LCD_GetYSize(), and BSP_LCD_InitEx().
# LCD Private Defines

| STM32F769I_DISCOVERY_LCD | Defines | Variables |
Defines

<table>
<thead>
<tr>
<th>Define</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCD_DSI_ID</td>
<td>0x11</td>
</tr>
<tr>
<td>LCD_DSI_ADDRESS</td>
<td>TS_I2C_ADDRESS</td>
</tr>
<tr>
<td>LCD_DSI_ID_REG</td>
<td>0xA8</td>
</tr>
</tbody>
</table>
Variables

| static DSI_VidCfgTypeDef hdsivideo_handle |
### Define Documentation

```
#define LCD_DSI_ADDRESS  TS_I2C_ADDRESS
```

Definition at line **104** of file `stm32f769i_discovery_lcd.c`.

Referenced by `LCD_IO_GetID()`.

```
#define LCD_DSI_ID  0x11
```

Definition at line **103** of file `stm32f769i_discovery_lcd.c`.

Referenced by `BSP_LCD_InitEx()`, and `LCD_IO_GetID()`.

```
#define LCD_DSI_ID_REG  0xA8
```

Definition at line **105** of file `stm32f769i_discovery_lcd.c`.

Referenced by `LCD_IO_GetID()`.
### Variable Documentation

**DSI_VidCfgTypeDef hdsivideo_handle [static]**

Definition at line **107** of file `stm32f769i_discovery_lcd.c`.

Referenced by `BSP_LCD_DisplayOff()`, `BSP_LCD_DisplayOn()`, and `BSP_LCD_InitEx()`.
# STM32F769I-Discovery BSP User Manual

## Directories

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
</table>

## TS Private Variables

- **STM32F769I_DISCOVERY TS**
### Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static TS_DrvTypeDef *</td>
<td>ts_driver</td>
<td></td>
</tr>
<tr>
<td>static uint8_t</td>
<td>ts_orientation</td>
<td></td>
</tr>
<tr>
<td>uint8_t</td>
<td>I2C_Address = 0</td>
<td></td>
</tr>
<tr>
<td>char *</td>
<td>ts_event_string_tab</td>
<td>Table for touchscreen event information display on LCD: table indexed on enum TS_TouchEventTypeDef information.</td>
</tr>
<tr>
<td></td>
<td>[TOUCH_EVENT_NB_MAX]</td>
<td></td>
</tr>
<tr>
<td>char *</td>
<td>ts_gesture_id_string_tab</td>
<td>Table for touchscreen gesture Id information display on LCD: table indexed on enum TS_GestureIdTypeDef information.</td>
</tr>
<tr>
<td></td>
<td>[GEST_ID_NB_MAX]</td>
<td></td>
</tr>
</tbody>
</table>
Variable Documentation

**uint8_t I2C_Address = 0**

Definition at line 120 of file stm32f769i_discovery_ts.c.

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

**TS_DrvTypeDef* ts_driver [static]**

Definition at line 118 of file stm32f769i_discovery_ts.c.

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

**char* ts_event_string_tab[TOUCH_EVENT_NB_MAX]**

Initial value:

```c
{  "None",
   "Press down",
   "Lift up",
   "Contact"
}
```

Table for touchscreen event information display on LCD: table indexed on enum TS_TouchEventTypeTypeDef information.

Definition at line 123 of file stm32f769i_discovery_ts.c.
char* ts_gesture_id_string_tab[GEST_ID_NB_MAX]

Initial value:

```c
{  "None",
     "Move Up",
     "Move Right",
     "Move Down",
     "Move Left",
     "Zoom In",
     "Zoom Out"
}
```

Table for touchscreen gesture Id information display on LCD : table indexed on enum TS_GestureIdTypeDef information.

Definition at line 130 of file stm32f769i_discovery_ts.c.

`uint8_t ts_orientation [static]`

Definition at line 119 of file stm32f769i_discovery_ts.c.

Referenced by BSP_TS_GetState(), and BSP_TS_Init().

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery
BSP User Manual by doxygen 1.7.6.1
 STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

STM32F769I_DISCOVERY_LOW_LEVEL Private Functions

STM32F769I_DISCOVERY LOW LEVEL
### Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>I2Cx_MspInit</code> *(I2C_HandleTypeDef <em>i2c_handler)</em></td>
<td>Initializes I2C MSP.</td>
</tr>
<tr>
<td><code>I2Cx_Init</code> *(I2C_HandleTypeDef <em>i2c_handler)</em></td>
<td>Initializes I2C HAL.</td>
</tr>
<tr>
<td><code>I2Cx_ReadMultiple</code> *(I2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t <em>Buffer, uint16_t Length)</em></td>
<td>Reads multiple data.</td>
</tr>
<tr>
<td><code>I2Cx_WriteMultiple</code> *(I2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t <em>Buffer, uint16_t Length)</em></td>
<td>Writes a value in a register of the device through BUS in using DMA mode.</td>
</tr>
<tr>
<td><code>I2Cx_IsDeviceReady</code> *(I2C_HandleTypeDef <em>i2c_handler, uint16_t DevAddress, uint32_t Trials)</em></td>
<td>Checks if target device is ready for communication.</td>
</tr>
<tr>
<td><code>I2Cx_Error</code> *(I2C_HandleTypeDef <em>i2c_handler, uint8_t Addr)</em></td>
<td>Manages error callback by re-initializing I2C.</td>
</tr>
</tbody>
</table>
Function Documentation

**static void I2Cx_Error** (I2C_HandleTypeDef *i2c_handler, uint8_t Addr) [static]

Manages error callback by re-initializing I2C.

**Parameters:**
- **i2c_handler**: I2C handler
- **Addr**: I2C Address

**Return values:**
- **None**

Definition at line 533 of file *stm32f769i_discovery.c*.

References **I2Cx_Init()**.

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

**static void I2Cx_Init** (I2C_HandleTypeDef *i2c_handler) [static]

Initializes I2C HAL.

**Parameters:**
- **i2c_handler**: I2C handler

**Return values:**
- **None**

Definition at line 435 of file *stm32f769i_discovery.c*.

References **DISCOVERY_AUDIO_I2Cx**, **DISCOVERY_EXT_I2Cx**, **hI2cAudioHandler**, and **I2Cx_MspInit()**.
Referenced by `AUDIO_IO_Init()`, `EEPROM_IO_Init()`, `HDMI_IO_Init()`, `I2Cx_Error()`, and `TS_IO_Init()`.

```c
static HAL_StatusTypeDef I2Cx_IsDeviceReady (I2C_HandleTypeDef i2c_handler,
                                          uint16_t DevAddress,
                                          uint32_t Trials)
```

Checks if target device is ready for communication.

**Note:**
This function is used with Memory devices

**Parameters:**
- `i2c_handler`: I2C handler
- `DevAddress`: Target device address
- `Trials`: Number of trials

**Return values:**
- HAL status

Definition at line 522 of file `stm32f769i_discovery.c`.

Referenced by `EEPROM_IO_IsDeviceReady()`.

```c
static void I2Cx_MspInit (I2C_HandleTypeDef * i2c_handler)
```

Initializes I2C MSP.

**Parameters:**
- `i2c_handler`: I2C handler

**Return values:**
- None
Definition at line 350 of file stm32f769i_discovery.c.

References DISCOVERY_AUDIO_I2Cx_CLK_ENABLE, DISCOVERY_AUDIO_I2Cx_ERIRQn, DISCOVERY_AUDIO_I2Cx_EVIRQn, DISCOVERY_AUDIO_I2Cx_FORCE_RESET, DISCOVERY_AUDIO_I2Cx_RELEASE_RESET, DISCOVERY_AUDIO_I2Cx_SCL_AF, DISCOVERY_AUDIO_I2Cx_SCL_GPIO_CLK_ENABLE, DISCOVERY_AUDIO_I2Cx_SCL_GPIO_PORT, DISCOVERY_AUDIO_I2Cx_SCL_PIN, DISCOVERY_AUDIO_I2Cx_SDA_AF, DISCOVERY_AUDIO_I2Cx_SDA_GPIO_CLK_ENABLE, DISCOVERY_AUDIO_I2Cx_SDA_GPIO_PORT, DISCOVERY_AUDIO_I2Cx_SDA_PIN, DISCOVERY_EXT_I2Cx_CLK_ENABLE, DISCOVERY_EXT_I2Cx_ERIRQn, DISCOVERY_EXT_I2Cx_EVIRQn, DISCOVERY_EXT_I2Cx_FORCE_RESET, DISCOVERY_EXT_I2Cx_RELEASE_RESET, DISCOVERY_EXT_I2Cx_SCL_PIN, DISCOVERY_EXT_I2Cx_SCL_SDA_AF, DISCOVERY_EXT_I2Cx_SCL_SDA_GPIO_CLK_ENABLE, DISCOVERY_EXT_I2Cx_SCL_SDA_GPIO_PORT, DISCOVERY_EXT_I2Cx_SCL_SDA_PIN, and hI2cAudioHandler.

Referenced by I2Cx_Init().

```c
static HAL_StatusTypeDef I2Cx_ReadMultiple ( I2C_HandleTypeDef *uint8_t *uint16_t *uint16_t *uint8_t *uint16_t )
```
Reads multiple data.

**Parameters:**
- `i2c_handler`: I2C handler
- `Addr`: I2C address
- `Reg`: Reg address
- `MemAddress`: memory address
- `Buffer`: Pointer to data buffer
- `Length`: Length of the data

**Return values:**
- `HAL` status

Definition at line 473 of file `stm32f769i_discovery.c`.

References `I2Cx_Error()`.

Referenced by `AUDIO_IO_Read()`, `EEPROM_IO_ReadData()`, `HDMI_IO_Read()`, `TS_IO_Read()`, and `TS_IO_ReadMultiple()`.

```c
static HAL_StatusTypeDef I2Cx_WriteMultiple ( I2C_HandleTypeDef * i2c_handler, uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t * Buffer, uint16_t Length )
```

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

**Parameters:**
- `i2c_handler`: I2C handler
- `Addr`: Device address on BUS Bus.
Reg, : The target register address to write
MemAddress, : memory address
Buffer, : The target register value to be written
Length, : buffer size to be written

Return values:
HAL status

Definition at line 499 of file stm32f769i_discovery.c.

References I2Cx_Error().

Referenced by AUDIO_IO_Write(), EEPROM_IO_WriteData(), HDMI_IO_Write(), TS_IO_Write(), and TS_IO_WriteMultiple().
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**STM32F769I Discovery**

**Low Level Led**

**STM32F769I Discovery Low Level Exported Constants**
#define LEDn ((uint8_t)2)
#define LED1_GPIO_PORT ((GPIO_TypeDef*)GPIOJ)
#define LED2_GPIO_PORT ((GPIO_TypeDef*)GPIOJ)
#define LEDx_GPIO_CLK_ENABLE() __HAL_RCC_GPIOJ_CLK_ENABLE()
#define LEDx_GPIO_CLK_DISABLE() __HAL_RCC_GPIOJ_CLK_DISABLE()
#define LED1_PIN ((uint32_t)GPIO_PIN_13)
#define LED2_PIN ((uint32_t)GPIO_PIN_5)
### Define Documentation

<table>
<thead>
<tr>
<th><strong>Define</strong></th>
<th><strong>Definition</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#define LED1_GPIO_PORT ((GPIO_TypeDef*)GPIOJ)</code></td>
<td>Definition at line 142 of file <code>stm32f769i_discovery.h</code>.</td>
</tr>
<tr>
<td><code>#define LED1_PIN ((uint32_t)GPIO_PIN_13)</code></td>
<td>Definition at line 148 of file <code>stm32f769i_discovery.h</code>.</td>
</tr>
<tr>
<td><code>#define LED2_GPIO_PORT ((GPIO_TypeDef*)GPIOJ)</code></td>
<td>Definition at line 143 of file <code>stm32f769i_discovery.h</code>.</td>
</tr>
<tr>
<td><code>#define LED2_PIN ((uint32_t)GPIO_PIN_5)</code></td>
<td>Definition at line 149 of file <code>stm32f769i_discovery.h</code>.</td>
</tr>
<tr>
<td><code>#define LEDn ((uint8_t)2)</code></td>
<td>Definition at line 138 of file <code>stm32f769i_discovery.h</code>.</td>
</tr>
<tr>
<td><code>#define LEDx_GPIO_CLK_DISABLE()</code></td>
<td>Definition at line 146 of file <code>stm32f769i_discovery.h</code>.</td>
</tr>
<tr>
<td><code>#define LEDx_GPIO_CLK_ENABLE()</code></td>
<td>Definition at line 145 of file <code>stm32f769i_discovery.h</code>.</td>
</tr>
</tbody>
</table>
Referenced by `BSP_LED_Init()`.
# STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## STM32F769I_DISCOVERY_QSPI Exported Constants

```
STM32F769I_DISCOVERY_QSPI
```

`STM32F769I_DISCOVERY_QSPI`
Defines

#define QSPI_OK        ((uint8_t)0x00)
#define QSPI_ERROR     ((uint8_t)0x01)
#define QSPI_BUSY      ((uint8_t)0x02)
#define QSPI_NOT_SUPPORTED ((uint8_t)0x04)
#define QSPI_SUSPENDED ((uint8_t)0x08)
#define QSPI_CLK_ENABLE()   __HAL_RCC_QSPI_CLK_ENABLE()
#define QSPI_CLK_DISABLE()  __HAL_RCC_QSPI_CLK_DISABLE()
#define QSPI_CS_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_ENABLE()
#define QSPI_CLK_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_ENABLE()
#define QSPI_D0_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define QSPI_D1_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define QSPI_D2_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define QSPI_D3_GPIO_CLK_ENABLE() __HAL_RCC_GPIOD_CLK_ENABLE()
#define QSPI_FORCE_RESET()   __HAL_RCC_QSPI_FORCE_RESET()
#define QSPI_RELEASE_RESET() __HAL_RCC_QSPI_RELEASE_RESET()
#define QSPI_CS_PIN        GPIO_PIN_6
#define QSPI_CS_GPIO_PORT  GPIOB
#define QSPI_CS_PIN_AF     GPIO_AF10_QUADSPI
#define QSPI_CLK_PIN       GPIO_PIN_2
#define QSPI_CLK_GPIO_PORT GPIOB
#define QSPI_CLK_PIN_AF    GPIO_AF9_QUADSPI
#define QSPI_D0_PIN        GPIO_PIN_9
#define QSPI_D0_GPIO_PORT  GPIOC
#define QSPI_D0_PIN_AF     GPIO_AF9_QUADSPI
#define QSPI_D1_PIN        GPIO_PIN_10
#define QSPI_D1_GPIO_PORT  GPIOC
#define QSPI_D1_PIN_AF     GPIO_AF9_QUADSPI
#define QSPI_D2_PIN        GPIO_PIN_2
#define QSPI_D2_GPIO_PORT  GPIOE
#define QSPI_D2_PIN_AF     GPIO_AF9_QUADSPI
#define QSPI_D3_PIN        GPIO_PIN_13
#define QSPI_D3_GPIO_PORT GPIOD
#define QSPI_D3_PIN_AF GPIO_AF9_QUADSPI
Define Documentation

```c
#define QSPI_BUSY  ((uint8_t)0x02)
```
Definition at line 71 of file `stm32f769i_discovery_qspi.h`.
Referenced by `BSP_QSPI_GetStatus()`.

```c
#define QSPI_CLK_DISABLE ( ) __HAL_RCC_QSPI_CLK_DISABLE()
```
Definition at line 78 of file `stm32f769i_discovery_qspi.h`.
Referenced by `BSP_QSPI_MspDeInit()`.

```c
#define QSPI_CLK_ENABLE ( ) __HAL_RCC_QSPI_CLK_ENABLE()
```
Definition at line 77 of file `stm32f769i_discovery_qspi.h`.
Referenced by `BSP_QSPI_MspInit()`.

```c
#define QSPI_CLK_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOB_CLK_ENABLE()
```
Definition at line 80 of file `stm32f769i_discovery_qspi.h`.
Referenced by `BSP_QSPI_MspInit()`.

```c
#define QSPI_CLK_GPIO_PORT GPIOB
```
Definition at line 96 of file `stm32f769i_discovery_qspi.h`.
Referenced by `BSP_QSPI_MspDeInit()`, and `BSP_QSPI_MspInit()`.
```c
#define QSPI_CLK_PIN GPIO_PIN_2

Definition at line 95 of file stm32f769i_discovery_qspi.h.
Referenced by BSP_QSPI_MspDeInit(), and BSP_QSPI_MspInit().

#define QSPI_CLK_PIN_AF GPIO_AF9_QUADSPI

Definition at line 97 of file stm32f769i_discovery_qspi.h.
Referenced by BSP_QSPI_MspInit().

#define QSPI_CS_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_ENABLE()

Definition at line 79 of file stm32f769i_discovery_qspi.h.
Referenced by BSP_QSPI_MspInit().

#define QSPI_CS_GPIO_PORT GPIOB

Definition at line 92 of file stm32f769i_discovery_qspi.h.
Referenced by BSP_QSPI_MspDeInit(), and BSP_QSPI_MspInit().

#define QSPI_CS_PIN GPIO_PIN_6

Definition at line 91 of file stm32f769i_discovery_qspi.h.
Referenced by BSP_QSPI_MspDeInit(), and BSP_QSPI_MspInit().

#define QSPI_CS_PIN_AF GPIO_AF10_QUADSPI
```
<table>
<thead>
<tr>
<th>Definition at line</th>
<th>Referenced by</th>
</tr>
</thead>
<tbody>
<tr>
<td>93 of file stm32f769i_discovery_qspi.h.</td>
<td>BSP_QSPI_MspInit().</td>
</tr>
<tr>
<td>#define QSPI_D0_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOC_CLK_ENABLE()</td>
<td></td>
</tr>
<tr>
<td>81 of file stm32f769i_discovery_qspi.h.</td>
<td>BSP_QSPI_MspInit().</td>
</tr>
<tr>
<td>#define QSPI_D0_GPIO_PORT GPIOC</td>
<td></td>
</tr>
<tr>
<td>100 of file stm32f769i_discovery_qspi.h.</td>
<td>BSP_QSPI_MspDeInit(), and BSP_QSPI_MspInit().</td>
</tr>
<tr>
<td>#define QSPI_D0_PIN GPIO_PIN_9</td>
<td></td>
</tr>
<tr>
<td>99 of file stm32f769i_discovery_qspi.h.</td>
<td>BSP_QSPI_MspDeInit(), and BSP_QSPI_MspInit().</td>
</tr>
<tr>
<td>#define QSPI_D0_PIN_AF GPIO_AF9_QUADSPI</td>
<td></td>
</tr>
<tr>
<td>101 of file stm32f769i_discovery_qspi.h.</td>
<td>BSP_QSPI_MspInit().</td>
</tr>
<tr>
<td>#define QSPI_D1_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOC_CLK_ENABLE()</td>
<td></td>
</tr>
<tr>
<td>82 of file stm32f769i_discovery_qspi.h.</td>
<td></td>
</tr>
</tbody>
</table>
Referenced by `BSP_QSPI_MspInit()`.

```c
#define QSPI_D1_GPIO_PORT GPIOC
```
Definition at line 104 of file `stm32f769i_discovery_qspi.h`.
Referenced by `BSP_QSPI_MspDeInit()`, and `BSP_QSPI_MspInit()`.

```c
#define QSPI_D1_PIN GPIO_PIN_10
```
Definition at line 103 of file `stm32f769i_discovery_qspi.h`.
Referenced by `BSP_QSPI_MspDeInit()`, and `BSP_QSPI_MspInit()`.

```c
#define QSPI_D1_PIN_AF GPIO_AF9_QUADSPI
```
Definition at line 105 of file `stm32f769i_discovery_qspi.h`.
Referenced by `BSP_QSPI_MspInit()`.

```c
#define QSPI_D2_GPIO_CLK_ENABLE () __HAL_RCC_GPIOE_CLK_ENABLE()
```
Definition at line 83 of file `stm32f769i_discovery_qspi.h`.
Referenced by `BSP_QSPI_MspInit()`.

```c
#define QSPI_D2_GPIO_PORT GPIOE
```
Definition at line 108 of file `stm32f769i_discovery_qspi.h`.
Referenced by `BSP_QSPI_MspDeInit()`, and `BSP_QSPI_MspInit()`.
<table>
<thead>
<tr>
<th>Definition</th>
<th>Referenced by</th>
</tr>
</thead>
<tbody>
<tr>
<td>#define QSPI_D2_PIN GPIO_PIN_2</td>
<td>BSP_QSPI_MspDeInit(), and BSP_QSPI_MspInit()</td>
</tr>
<tr>
<td>Definition at line 107 of file stm32f769i_discovery_qspi.h.</td>
<td></td>
</tr>
<tr>
<td>#define QSPI_D2_PIN_AF GPIO_AF9_QUADSPI</td>
<td>BSP_QSPI_MspInit()</td>
</tr>
<tr>
<td>Definition at line 109 of file stm32f769i_discovery_qspi.h.</td>
<td></td>
</tr>
<tr>
<td>#define QSPI_D3_GPIO_CLK_ENABLE __HAL_RCC_GPIOD_CLK_ENABLE()</td>
<td>BSP_QSPI_MspInit()</td>
</tr>
<tr>
<td>Definition at line 84 of file stm32f769i_discovery_qspi.h.</td>
<td></td>
</tr>
<tr>
<td>#define QSPI_D3_GPIO_PORT GPIOD</td>
<td>BSP_QSPI_MspDeInit(), and BSP_QSPI_MspInit()</td>
</tr>
<tr>
<td>Definition at line 112 of file stm32f769i_discovery_qspi.h.</td>
<td></td>
</tr>
<tr>
<td>#define QSPI_D3_PIN GPIO_PIN_13</td>
<td>BSP_QSPI_MspDeInit(), and BSP_QSPI_MspInit()</td>
</tr>
<tr>
<td>Definition at line 111 of file stm32f769i_discovery_qspi.h.</td>
<td></td>
</tr>
<tr>
<td>#define QSPI_D3_PIN_AF GPIO_AF9_QUADSPI</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
#define QSPI_ERROR (uint8_t)0x01

Definition at line 113 of file stm32f769i_discovery_qspi.h.

Referenced by BSP_QSPI_MspInit().

#define QSPI_FORCE_RESET ( ) __HAL_RCC_QSPI_FORCE_RESET()

Definition at line 86 of file stm32f769i_discovery_qspi.h.

Referenced by BSP_QSPI_MspDeInit(), and BSP_QSPI_MspInit().

#define QSPI_NOT_SUPPORTED (uint8_t)0x04

Definition at line 72 of file stm32f769i_discovery_qspi.h.

Referenced by BSP_QSPI_DeInit(), and BSP_QSPI_Init().

#define QSPI_OK (uint8_t)0x00

Definition at line 70 of file stm32f769i_discovery_qspi.h.

Referenced by BSP_QSPI_MspDeInit(), BSP_QSPI_MspInit(),
BSP_QSPI_EnableMemoryMappedMode(),
BSP_QSPI_Erase_Block(), BSP_QSPI_Erase_Chip(),
BSP_QSPI_GetStatus(), BSP_QSPI_Init(), BSP_QSPI_Read(),
BSP_QSPI_Write(), QSPI_AutoPollingMemReady(),
QSPI_DummyCyclesCfg(), QSPI_EnterFourBytesAddress(),
QSPI_EnterMemory_QPI(), QSPI_ExitMemory_QPI(),
QSPI_OutDrvStrengthCfg(), QSPI_ResetMemory(), and
QSPI_WriteEnable().
Definition at line 69 of file stm32f769i_discovery_qspi.h.

Referenced by BSP_QSPI_DeInit(),
BSP_QSPI_EnableMemoryMappedMode(),
BSP_QSPI_Erase_Block(), BSP_QSPI_Erase_Chip(),
BSP_QSPI_GetInfo(), BSP_QSPI_GetStatus(), BSP_QSPI_Init(),
BSP_QSPI_Read(), BSP_QSPI_Write(),
QSPI_AutoPollingMemReady(), QSPI_DummyCyclesCfg(),
QSPI_EnterFourBytesAddress(), QSPI_EnterMemory_QPI(),
QSPI_ExitMemory_QPI(), QSPI_OutDrvStrengthCfg(),
QSPI_ResetMemory(), and QSPI_WriteEnable().

#define QSPI_RELEASE_RESET ( ) __HAL_RCC_QSPI_RELEASE_RESET

Definition at line 87 of file stm32f769i_discovery_qspi.h.

Referenced by BSP_QSPI_MspDeInit(), and BSP_QSPI_MspInit().

#define QSPI_SUSPENDED ( (uint8_t)0x08)

Definition at line 73 of file stm32f769i_discovery_qspi.h.
## STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### STM32F769I_DISCOVERY QSPI Private Variables

**STM32F769I_DISCOVERY QSPI**
## Variables

| QSPI_HandleTypeDef | QSPIHandle |
Variable Documentation

QSPI_HandleTypeDef QSPIHandle

Definition at line 88 of file stm32f769i_discovery_qspi.c.

Referenced by BSP_QSPI_DeInit(),
BSP_QSPI_EnableMemoryMappedMode(),
BSP_QSPI_Erase_Block(), BSP_QSPI_Erase_Chip(),
BSP_QSPI_GetStatus(), BSP_QSPI_Init(), BSP_QSPI_Read(),
BSP_QSPI_Write(), and QSPI_OutDrvStrengthCfg().

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery
BSP User Manual by doxygen 1.7.6.1
# STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## SDRAM Exported Types

**STM32F769I_DISCOVERY SDRAM**
Modules

SDRAM Exported Constants
### Defines

<table>
<thead>
<tr>
<th>Define</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SDRAM_OK</code></td>
<td>((uint8_t)0x00)</td>
<td>SDRAM status structure definition.</td>
</tr>
<tr>
<td><code>SDRAM_ERROR</code></td>
<td>((uint8_t)0x01)</td>
<td></td>
</tr>
<tr>
<td><code>SDRAM_MODEREG_BURST_LENGTH_1</code></td>
<td>((uint16_t)0x0000)</td>
<td>FMC SDRAM Mode definition register defines.</td>
</tr>
<tr>
<td><code>SDRAM_MODEREG_BURST_LENGTH_2</code></td>
<td>((uint16_t)0x0001)</td>
<td></td>
</tr>
<tr>
<td><code>SDRAM_MODEREG_BURST_LENGTH_4</code></td>
<td>((uint16_t)0x0002)</td>
<td></td>
</tr>
<tr>
<td><code>SDRAM_MODEREG_BURST_LENGTH_8</code></td>
<td>((uint16_t)0x0004)</td>
<td></td>
</tr>
<tr>
<td><code>SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL</code></td>
<td>((uint16_t)0x0000)</td>
<td></td>
</tr>
<tr>
<td><code>SDRAM_MODEREG_BURST_TYPE_INTERLEAVED</code></td>
<td>((uint16_t)0x0008)</td>
<td></td>
</tr>
<tr>
<td><code>SDRAM_MODEREG_CAS_LATENCY_2</code></td>
<td>((uint16_t)0x0020)</td>
<td></td>
</tr>
<tr>
<td><code>SDRAM_MODEREG_CAS_LATENCY_3</code></td>
<td>((uint16_t)0x0030)</td>
<td></td>
</tr>
<tr>
<td><code>SDRAM_MODEREG_OPERATING_MODE_STANDARD</code></td>
<td>((uint16_t)0x0000)</td>
<td></td>
</tr>
<tr>
<td><code>SDRAM_MODEREG_WRITEBURST_MODE_PROGRAMMED</code></td>
<td>((uint16_t)0x0200)</td>
<td></td>
</tr>
<tr>
<td><code>SDRAM_MODEREG_WRITEBURST_MODE_SINGLE</code></td>
<td>((uint16_t)0x0000)</td>
<td></td>
</tr>
</tbody>
</table>
#define SDRAM_ERROR ((uint8_t)0x01)

Definition at line 70 of file stm32f769i_discovery_sdram.h.

Referenced by BSP_SDRAM_DeInit(), BSP_SDRAM_Init(), BSP_SDRAM_ReadData(), BSP_SDRAM_ReadData_DMA(), BSP_SDRAM_Sendcmd(), BSP_SDRAM_WriteData(), and BSP_SDRAM_WriteData_DMA().

#define SDRAM_MODEREG_BURST_LENGTH_1 ((uint16_t)0x0000)

FMC SDRAM Mode definition register defines.

Definition at line 104 of file stm32f769i_discovery_sdram.h.

Referenced by BSP_SDRAM_Initialization_sequence().

#define SDRAM_MODEREG_BURST_LENGTH_2 ((uint16_t)0x0001)

Definition at line 105 of file stm32f769i_discovery_sdram.h.

#define SDRAM_MODEREG_BURST_LENGTH_4 ((uint16_t)0x0002)

Definition at line 106 of file stm32f769i_discovery_sdram.h.

#define SDRAM_MODEREG_BURST_LENGTH_8 ((uint16_t)0x0004)

Definition at line 107 of file stm32f769i_discovery_sdram.h.
```c
#define SDRAM_MODEREG_BURST_TYPE_INTERLEAVED ((uint16_t)0x0008)

Definition at line 109 of file stm32f769i_discovery_sdram.h.

#define SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL ((uint16_t)0x0000)

Definition at line 108 of file stm32f769i_discovery_sdram.h.

Referenced by BSP_SDRAM_Initialization_sequence().

#define SDRAM_MODEREG_CAS_LATENCY_2 ((uint16_t)0x0020)

Definition at line 110 of file stm32f769i_discovery_sdram.h.

#define SDRAM_MODEREG_CAS_LATENCY_3 ((uint16_t)0x0030)

Definition at line 111 of file stm32f769i_discovery_sdram.h.

Referenced by BSP_SDRAM_Initialization_sequence().

#define SDRAM_MODEREG_OPERATING_MODE_STANDARD ((uint16_t)0x0000)

Definition at line 112 of file stm32f769i_discovery_sdram.h.

Referenced by BSP_SDRAM_Initialization_sequence().

#define SDRAM_MODEREG_WRITEBURST_MODE_PROGRAMMED

Definition at line 113 of file stm32f769i_discovery_sdram.h.

#define SDRAM_MODEREG_WRITEBURST_MODE_SINGLE ((uint16_t)0x0200)
```
Definition at line 114 of file stm32f769i_discovery_sdram.h.

Referenced by BSP_SDRAM_Initialization_sequence().

```c
#define SDRAM_OK  ((uint8_t)0x00)
```

SDRAM status structure definition.

Definition at line 69 of file stm32f769i_discovery_sdram.h.

Referenced by BSP_SDRAM_DeInit(), BSP_SDRAM_Init(), BSP_SDRAM_ReadData(), BSP_SDRAM_ReadData_DMA(), BSP_SDRAM_Sendcmd(), BSP_SDRAM_WriteData(), and BSP_SDRAM_WriteData_DMA().

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
# TS Imported Variables

<table>
<thead>
<tr>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F769I_DISCOVERY TS</td>
</tr>
</tbody>
</table>
Variables

| char * ts_event_string_tab [TOUCH_EVENT_NB_MAX] | Table for touchscreen event information display on LCD: table indexed on enum TS_TouchEventTypeDef information. |
| char * ts_gesture_id_string_tab [GEST_ID_NB_MAX] | Table for touchscreen gesture Id information display on LCD: table indexed on enum TS_GestureIdTypeDef information. |
### Variable Documentation

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
<th>Definition Location</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>char* ts_event_string_tab[TOUCH_EVENT_NB_MAX]</code></td>
<td>Table for touchscreen event information display on LCD: table indexed on enum <code>TS_TouchEventTypeDef</code> information.</td>
<td>Definition at line 123 of file <code>stm32f769i_discovery_ts.c</code>.</td>
</tr>
<tr>
<td><code>char* ts_gesture_id_string_tab[GEST_ID_NB_MAX]</code></td>
<td>Table for touchscreen gesture Id information display on LCD: table indexed on enum <code>TS_GestureIdTypeDef</code> information.</td>
<td>Definition at line 130 of file <code>stm32f769i_discovery_ts.c</code>.</td>
</tr>
<tr>
<td>Main Page</td>
<td>Modules</td>
<td>Data Structures</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Directories</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**STM32F769I Discovery**

**Sd Private Variables**

STM32F769I_DISCOVERY SD
## Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD_HandleTypeDef</td>
<td>uSdHandle</td>
</tr>
</tbody>
</table>
Variable Documentation

**SD_HandleTypeDef uSdHandle**

Definition at line 120 of file stm32f769i_discovery_sd.c.

Referenced by **BSP_SD_DeInit()**, **BSP_SD_Erase()**, **BSP_SD_GetCardInfo()**, **BSP_SD_GetCardState()**, **BSP_SD_Init()**, **BSP_SD_ReadBlocks()**, **BSP_SD_ReadBlocks_DMA()**, **BSP_SD_WriteBlocks()**, and **BSP_SD_WriteBlocks_DMA()**.
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Drivers</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Drivers Directory Reference**
## Directories

<table>
<thead>
<tr>
<th>directory</th>
<th>BSP</th>
</tr>
</thead>
</table>

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery
BSP User Manual by [doxygen](http://www.doxygen.org) 1.7.6.1
STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Drivers</td>
<td>BSP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

BSP Directory Reference
## Directories

<table>
<thead>
<tr>
<th>directory</th>
<th>STM32F769I-Discovery</th>
</tr>
</thead>
</table>

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery
BSP User Manual by doxygen 1.7.6.1
STM32F769I-Discovery BSP User Manual

STM32F769I-Discovery Directory Reference
**Files**

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>stm32f769i_discovery.c</strong></td>
<td>This file provides a set of firmware functions to manage LEDs, push-buttons, external SDRAM, external QSPI Flash, RF EEPROM, available on STM32F769I-Discovery board (MB1225) from STMicroelectronics.</td>
</tr>
<tr>
<td><strong>stm32f769i_discovery.h</strong></td>
<td>This file contains definitions for STM32F769I-Discovery LEDs, push-buttons hardware resources.</td>
</tr>
<tr>
<td><strong>stm32f769i_discovery_audio.c</strong></td>
<td>This file provides the Audio driver for the STM32F769I-DISCOVERY board.</td>
</tr>
<tr>
<td><strong>stm32f769i_discovery_audio.h</strong></td>
<td>This file contains the common defines and functions prototypes for the <strong>stm32f769i_discovery_audio.c</strong> driver.</td>
</tr>
<tr>
<td><strong>stm32f769i_discovery_eeprom.c</strong></td>
<td>This file provides a set of functions needed to manage an I2C M24LR64 EEPROM memory.</td>
</tr>
<tr>
<td><strong>stm32f769i_discovery_eeprom.h</strong></td>
<td>This file contains all the functions prototypes for the</td>
</tr>
</tbody>
</table>
**stm32f769i_discovery_eeprom.c** firmware driver.

**stm32f769i_discovery_lcd.c** [code]

This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM32F769I-DISCOVERY board.

**stm32f769i_discovery_lcd.h** [code]

This file contains the common defines and functions prototypes for the stm32f769i_discovery_lcd.c driver.

**stm32f769i_discovery_qspi.c** [code]

This file includes a standard driver for the MX25L512 QSPI memory mounted on STM32F769I-Discovery board.

**stm32f769i_discovery_qspi.h** [code]

This file contains the common defines and functions prototypes for the **stm32f769i_discovery_qspi.c** driver.

**stm32f769i_discovery_sd.c** [code]

This file includes the uSD card driver mounted on STM32F769I-Discovery board.

**stm32f769i_discovery_sd.h** [code]

This file contains the common defines and functions prototypes for the **stm32f769i_discovery_sd.c** driver.
This file includes the SDRAM driver for the MT48LC4M32B2B5-6A memory device mounted on STM32F769I-DISCOVERY boards.

This file contains the common defines and functions prototypes for the `stm32f769i_discovery_sdram.c` driver.

This file provides a set of functions needed to manage the Touch Screen on STM32F769I-DISCOVERY discovery board.

This file contains the common defines and functions prototypes for the `stm32f769i_discovery_ts.c` driver.
Go to the documentation of this file.

```c
/**
 * @file    stm32f769i_discovery.h
 * @author  MCD Application Team
 * @version V2.0.0
 * @date    30-December-2016
 * @brief   This file contains definitions for STM32F769I-Discovery LEDs,
 *          push-buttons hardware resources.

 * @attention
 *
 * <h2><center>&copy; COPYRIGHT(c) 2016 STM
 *   microelectronics</center></h2>
 * @
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
```
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
/* Define to prevent recursive inclusion -----------------------*/
#ifndef __STM32F769I_DISCOVERY_H
#define __STM32F769I_DISCOVERY_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes -----------------------------------------------*/
#include "stm32f7xx_hal.h"

/** @addtogroup BSP *
 * @{
 */

/** @addtogroup STM32F769I_DISCOVERY *
 * @{
 */

/** @defgroup STM32F769I_DISCOVERY_LOW_LEVEL STM32F769I-Discovery LOW LEVEL *
 * @{
 */

/** @defgroup STM32F769I_DISCOVERY_LOW_LEVEL_Exported_Types STM32F769I Discovery Low Level Exported Types *
 * @{
 */
/**
 * @brief Define for STM32F769I_DISCOVERY board
 */
#if !defined (USE_STM32F769I_DISCO)
#define USE_STM32F769I_DISCO
#endif

/**
 * @brief Led_TypeDef
 * STM32F769I_DISCOVERY board leds definitions.
 */
typedef enum
{
    LED1 = 0,
    LED_RED = LED1,
    LED2 = 1,
    LED_GREEN = LED2
} Led_TypeDef;

/**
 * @brief Button_TypeDef
 * STM32F769I_DISCOVERY board Buttons definitions.
 */
typedef enum
{
    BUTTON_WAKEUP = 0,
} Button_TypeDef;

#define BUTTON_USER BUTTON_WAKEUP

/**
 * @brief ButtonMode_TypeDef
 * STM32F769I_DISCOVERY board Buttons Modes definitions.
 */
typedef enum
BUTTON_MODE_GPIO = 0,
BUTTON_MODE_EXTI = 1
}

ButtonMode_TypeDef;

/**	@addtogroup	Exported_types
@{
*/
typedef enum
{
    PB_SET = 0,
    PB_RESET = !PB_SET
} ButtonValue_TypeDef;

/**	@brief	DISCO_Status_TypeDef
* STM32F769I_DISCO board Status return possible values.
*/
typedef enum
{
    DISCO_OK = 0,
    DISCO_ERROR = 1
} DISCO_Status_TypeDef;

/**	@}
*/

/**	@defgroup	STM32F769I_DISCOVERY_LOW_LEVEL_Exported_Constants	STM32F769I Discovery Low Level Exported Constants
@{
*/
/**
 * @addtogroup STM32F769I_DISCOVERY_LOW_LEVEL_LED
 * STM32F769I Discovery Low Level Led
 *
 */
/*
 * Always four leds for all revisions of Discovery boards */
#define LEDn ((uint8_t)2)

/*
2 Leds are connected to MCU directly on PJ13 and PJ5 */
#define LED1_GPIO_PORT ((GPIO_TypeDef*)GPIOJ)
#define LED2_GPIO_PORT ((GPIO_TypeDef*)GPIOJ)
#define LEDx_GPIO_CLK_ENABLE() __HAL_RCC_GPIOJ_CLK_ENABLE()
#define LEDx_GPIO_CLK_DISABLE() __HAL_RCC_GPIOJ_CLK_DISABLE()
#define LED1_PIN ((uint32_t)GPIO_PIN_13)
#define LED2_PIN ((uint32_t)GPIO_PIN_5)

/**
 * @addtogroup STM32F769I_DISCOVERY_LOW_LEVEL_BUTTON
 * STM32F769I Discovery Low Level Button
 *
 */
/*
Only one User/Wakeup button */
#define BUTTONn ((uint8_t)1)

/**
 * @brief Wakeup push-button
 *
 */
#define WAKEUP_BUTTON_PIN GPIO_PIN_0
#define WAKEUP_BUTTON_GPIO_PORT GPIOA
#define WAKEUP_BUTTON_GPIO_CLK_ENABLE() __HAL_RCC_GPIOA_CLK_ENABLE()
#define WAKEUP_BUTTON_GPIO_CLK_DISABLE() __HAL_RCC_GPIOA_CLK_DISABLE()
#define WAKEUP_BUTTON_EXTI_IRQn EXTI0_IRQn

/* Define the USER button as an alias of the Wakeup button */
#define USER_BUTTON_PIN WAKEUP_BUTTON_PIN
#define USER_BUTTON_GPIO_PORT WAKEUP_BUTTON_GPIO_PORT
#define USER_BUTTON_GPIO_CLK_ENABLE() WAKEUP_BUTTON_GPIO_CLK_ENABLE()
#define USER_BUTTON_GPIO_CLK_DISABLE() WAKEUP_BUTTON_GPIO_CLK_DISABLE()
#define USER_BUTTON_EXTI_IRQn WAKEUP_BUTTON_EXTI_IRQn

#define BUTTON_GPIO_CLK_ENABLE() __HAL_RCC_GPIOA_CLK_ENABLE()

/**
 * @}
 */

/**
 * @brief USB OTG HS Over Current signal
 */
#define OTG_HS_OVER_CURRENT_PIN GPIO_PIN_4
#define OTG_HS_OVER_CURRENT_PORT GPIOD
#define OTG_HS_OVER_CURRENT_PORT_CLK_ENABLE(__HAL_RCC_GPIOD_CLK_ENABLE())

/**
 * @brief SD-detect signal
 */
#define SD_DETECT_PIN ((uint32_t)GPIO_PIN_15)
#define SD_DETECT_GPIO_PORT ((GPIO_TypeDef*)GPIOI)
#define SD_DETECT_GPIO_CLK_ENABLE(__HAL_RCC_GPIOI_CLK_ENABLE())
#define SD_DETECT_GPIO_CLK_DISABLE(__HAL_RCC_GPIOI_CLK_DISABLE())
#define SD_DETECT_EXTI_IRQn EXTI15_10_IRQn

/**
 * @brief Touch screen interrupt signal
 */
#define TS_INT_PIN ((uint32_t)GPIO_PIN_13)
#define TS_INT_GPIO_PORT ((GPIO_TypeDef*)GPIOI)
#define TS_INT_GPIO_CLK_ENABLE(__HAL_RCC_GPIOI_CLK_ENABLE())
#define TS_INT_GPIO_CLK_DISABLE(__HAL_RCC_GPIOI_CLK_DISABLE())
#define TS_INT_EXTI_IRQn EXTI15_10_IRQn
/**
 * @brief TouchScreen FT6206 Slave I2C address 1
 */
#define TS_I2C_ADDRESS ((uint16_t)0x54)

/**
 * @brief TouchScreen FT6336G Slave I2C address 2
 */
#define TS_I2C_ADDRESS_A02 ((uint16_t)0x70)

/**
 * @brief Audio I2C Slave address
 */
#define AUDIO_I2C_ADDRESS ((uint16_t)0x34)

/**
 * @brief EEPROM I2C Slave address 1
 */
#define EEPROM_I2C_ADDRESS_A01 ((uint16_t)0xA0)

/**
 * @brief EEPROM I2C Slave address 2
 */
#define EEPROM_I2C_ADDRESS_A02 ((uint16_t)0xA6)

/**
 * @brief User can use this section to tailor I2C4/I2C4 instance used and associated resources (audio codec).
 */
* Definition for I2C4 clock resources

```c
#define DISCOVERY_AUDIO_I2Cx
   I2C4
```

```c
#define DISCOVERY_AUDIO_I2Cx_CLK_ENABLE()
   __HAL_RCC_I2C4_CLK_ENABLE()
```

```c
#define DISCOVERY_AUDIO_I2Cx_SCL_GPIO_CLK_EN
   __HAL_RCC_GPIOD_CLK_ENABLE()
```

```c
#define DISCOVERY_AUDIO_I2Cx_SDA_GPIO_CLK_EN
   __HAL_RCC_GPIOB_CLK_ENABLE()
```

```c
#define DISCOVERY_AUDIO_I2Cx_FORCE_RESET()
   __HAL_RCC_I2C4_FORCE_RESET()
```

```c
#define DISCOVERY_AUDIO_I2Cx_RELEASE_RESET()
   __HAL_RCC_I2C4_RELEASE_RESET()
```

```c
/**
@brief Definition for I2C4 Pins

```c
#define DISCOVERY_AUDIO_I2Cx_SCL_PIN
   GPIO_PIN_12 /*!< PD12 */
```

```c
#define DISCOVERY_AUDIO_I2Cx_SCL_AF
   GPIO_AF4_I2C4
```

```c
#define DISCOVERY_AUDIO_I2Cx_SCL_GPIO_PORT
   GPIOD
```

```c
#define DISCOVERY_AUDIO_I2Cx_SDA_PIN
   GPIO_PIN_7 /*!< PB7 */
```

```c
#define DISCOVERY_AUDIO_I2Cx_SDA_AF
   GPIO_AF11_I2C4
```

```c
#define DISCOVERY_AUDIO_I2Cx_SDA_GPIO_PORT
   GPIOB
```

```c
/**
@brief Definition of I2C4 interrupt requests
```

```c
#define DISCOVERY_AUDIO_I2Cx_EV_IRQn
   I2C4_EV_IRQn
```

```c
#define DISCOVERY_AUDIO_I2Cx_ER_IRQn
   I2C4_ER_IRQn
```
User can use this section to tailor or I2C1/I2C1 instance used and associated resources.

Definition for I2C1 clock resources

```c
#define DISCOVERY_EXT_I2Cx
    I2C1
#define DISCOVERY_EXT_I2Cx_CLK_ENABLE() __HAL_RCC_I2C1_CLK_ENABLE()
#define DISCOVERY_DMAx_CLK_ENABLE() __HAL_RCC_DMA1_CLK_ENABLE()
#define DISCOVERY_EXT_I2Cx_SCL_SDA_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_ENABLE()
#define DISCOVERY_EXT_I2Cx_FORCE_RESET() __HAL_RCC_I2C1_FORCE_RESET()
#define DISCOVERY_EXT_I2Cx_RELEASE_RESET() __HAL_RCC_I2C1_RELEASE_RESET()
```

### Definition for I2C1 Pins

```c
#define DISCOVERY_EXT_I2Cx_SCL_PIN_GPIO_PIN_8 /*!< PB8 */
#define DISCOVERY_EXT_I2Cx_SCL_SDA_GPIO_PORT_GPIOB
#define DISCOVERY_EXT_I2Cx_SCL_SDA_AF_GPIO_AF4_I2C1
#define DISCOVERY_EXT_I2Cx_SDA_PIN_GPIO_PIN_9 /*!< PB9 */
```

### Definition of I2C interrupt requests

```c
#define DISCOVERY_EXT_I2Cx_EV_IRQn I2C1_EVIRQn
```
```c
#define DISCOVERY_EXT_I2Cx_ER_IRQn
#define I2C1_ER_IRQn

/* I2C TIMING Register define when I2C clock source is SYSCLK */
/* I2C TIMING is calculated from APB1 source clock = 50 MHz */
/* Due to the big MOFSET capacity for adapting the camera level the rising time is very large (>1us) */
/* 0x40912732 takes in account the big rising and aims a clock of 100khz */
#ifndef DISCOVERY_I2Cx_TIMING
#define DISCOVERY_I2Cx_TIMING
  ((uint32_t)0x40912732)
#endif /* DISCOVERY_I2Cx_TIMING */

/** @}
 * @} */

/**
 * @defgroup STM32F769I_DISCOVERY_LOW_LEVEL_Exported_Macros STM32F769I Discovery Low Level Exported Macros
 * @{ */

/* @} */

/**
 * @defgroup STM32F769I_DISCOVERY_LOW_LEVEL_Exported_Functions STM32F769I Discovery Low Level Exported Functions
 * @{ */

uint32_t BSP_GetVersion(void);
*/```
void BSP_LED_Init(Led_TypeDef Led);
void BSP_LED_DeInit(Led_TypeDef Led);
void BSP_LED_On(Led_TypeDef Led);
void BSP_LED_Off(Led_TypeDef Led);
void BSP_LED_Toggle(Led_TypeDef Led);
void BSP_PB_Init(Button_TypeDef Button, ButtonMode_TypeDef Button_Mode);
void BSP_PB_DeInit(Button_TypeDef Button);
uint32_t BSP_PB_GetState(ButtonTypeDef Button);
/**
 * @}
 */
/**
 * @}
 */
/**
 * @}
 */
/**
 * @}
 */
/**
 * @}
 */
/**
 * @}
 */
/**
 * @}
 */
#ifdef __cplusplus
}
#endif
#endif
#endif /* __STM32F769I_DISCOVERY_H */
/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
stm32f769i_discovery.c

Go to the documentation of this file.

00001  /**
00002  ******************************************************
00003  * @file      stm32f769i_discovery.c
00004  * @author    MCD Application Team
00005  * @version   V2.0.0
00006  * @date      30-December-2016
00007  * @brief     This file provides a set of firmware functions to manage LEDs,
00008  *            push-buttons, external SDRAM, external QSPI Flash, RF EEPROM,
00009  *            available on STM32F769I-Discovery board (MB1225) from
00010  *            STMicroelectronics.
00011  ******************************************************
00012  * @attention
00013  *
00014  * <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>
00015  *
00016  * Redistribution and use in source and bin
ary forms, with or without modification,
* are permitted provided that the following conditions are met:
  * 1. Redistributions of source code must retain the above copyright notice,
  * 2. Redistributions in binary form must reproduce the above copyright notice,
  * 3. Neither the name of STMicroelectronics nor the names of its contributors
  * may be used to endorse or promote products derived from this software
  * without specific prior written permission.

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

*/

#include "stm32f769i_discovery.h"

/** * @addtogroup BSP
 @{
 *
 */

/** * @addtogroup STM32F769I_DISCOVERY
 @{
 *
 */

/** * @defgroup STM32F769I_DISCOVERY_LOW_LEVEL
 STM32F769I_DISCOVERY LOW LEVEL
 *
 */

/** * @defgroup STM32F769I_DISCOVERY_LOW_LEVEL_Private_TypesDefinitions
 STM32F769I Discovery Low Level Private Typedef
 *
 */

/** * @defgroup STM32F769I_DISCOVERY_LOW_LEVEL_Private_Defines
 LOW_LEVEL Private Defines
 *
 */
* @brief STM32F769I Discovery BSP Driver version number V2.0.0 *

#define __STM32F769I_DISCOVERY_BSP_VERSION_MAIN (0x02) /*!< [31:24] main version */
#define __STM32F769I_DISCOVERY_BSP_VERSION_SUB1 (0x00) /*!< [23:16] sub1 version */
#define __STM32F769I_DISCOVERY_BSP_VERSION_SUB2 (0x00) /*!< [15:8] sub2 version */
#define __STM32F769I_DISCOVERY_BSP_VERSION_RC (0x00) /*!< [7:0] release candidate */

#define __STM32F769I_DISCOVERY_BSP_VERSION ((__STM32F769I_DISCOVERY_BSP_VERSION_MAIN << 24)\
  |(__STM32F769I_DISCOVERY_BSP_VERSION_SUB1 << 16)\
  |(__STM32F769I_DISCOVERY_BSP_VERSION_SUB2 << 8 )\
  |(__STM32F769I_DISCOVERY_BSP_VERSION_RC))

* @}*/

/** @defgroup STM32F769I_DISCOVERY_LOW_LEVEL_Private_Macros LOW_LEVEL Private Macros */
* @}*/

/** @defgroup STM32F769I_DISCOVERY_LOW_LEVEL_Private_Macros */
`Private_Variables` LOW_LEVEL Private Variables

```c
* @{
uint32_t GPIO_PIN[LEDn] = {LED1_PIN, LED2_PIN};

GPIO_TypeDef* GPIO_PORT[LEDn] = {LED1_GPIO_PORT, LED2_GPIO_PORT};

GPIO_TypeDef* BUTTON_PORT[BUTTONn] = {WAKEUP_BUTTON_GPIO_PORT};

const uint16_t BUTTON_PIN[BUTTONn] = {WAKEUP_BUTTON_PIN};

const uint16_t BUTTON_IRQn[BUTTONn] = {WAKEUP_BUTTON_EXTI_IRQn};

static I2C_HandleTypeDef hI2cAudioHandler = {0};

static I2C_HandleTypeDef hI2cExtHandler = {0};

/**
 * @{
 */

/**
 * @defgroup STM32F769I_DISCOVERY_LOW_LEVEL_Private_FunctionPrototypes LOW_LEVEL Private FunctionPrototypes
 * @{
 */

static void I2Cx_MspInit(I2C_HandleTypeDef *i2c_handler);
```
0115 static void I2Cx_Init(I2C_HandleTypeDef *i2c_handler);
0116
0117 static HAL_StatusTypeDef I2Cx_ReadMultiple(I2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint16_t Reg, uint16_t MemAddSize, uint8_t *Buffer, uint16_t Length);
0118 static HAL_StatusTypeDef I2Cx_WriteMultiple(I2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint16_t Reg, uint16_t MemAddSize, uint8_t *Buffer, uint16_t Length);
0119 static HAL_StatusTypeDef I2Cx_IsDeviceReady(I2C_HandleTypeDef *i2c_handler, uint16_t DevAddress, uint32_t Trials);
0120 static void I2Cx_Error(I2C_HandleTypeDef *i2c_handler, uint8_t Addr);
0121
0122 /* AUDIO IO functions */
0123 void AUDIO_IO_Init(void);
0124 void AUDIO_IO_DeInit(void);
0125 void AUDIO_IO_Write(uint8_t Addr, uint16_t Reg, uint16_t Value);
0126 uint16_t AUDIO_IO_Read(uint8_t Addr, uint16_t Reg);
0127 void AUDIO_IO_Delay(uint32_t Delay);
0128
0129 /* HDMI IO functions */
0130 void HDMI_IO_Init(void);
0131 void HDMI_IO_Delay(uint32_t Delay);
0132 void HDMI_IO_Write(uint8_t Addr, uint8_t Reg, uint8_t Value);
0133 uint8_t HDMI_IO_Read(uint8_t Addr, uint8_t Reg);
0134
0135 /* I2C EEPROM IO function */
void EEPROM_IO_Init(void);

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

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

HAL_StatusTypeDef EEPROM_IO_IsDeviceReady(uint16_t DevAddress, uint32_t Trials);

/*
  TouchScreen (TS) IO functions */

void TS_IO_Init(void);

void TS_IO_Write(uint8_t Addr, uint8_t Reg, uint8_t Value);

uint8_t TS_IO_Read(uint8_t Addr, uint8_t Reg);

uint16_t TS_IO_ReadMultiple(uint8_t Addr, uint8_t Reg, uint8_t* Buffer, uint16_t Length);

void TS_IO_WriteMultiple(uint8_t Addr, uint8_t Reg, uint8_t* Buffer, uint16_t Length);

void TS_IO_Delay(uint32_t Delay);

/* LCD Display IO functions */

void OTM8009A_IO_Delay(uint32_t Delay);

/**
  * @}
  */

/**
 * @defgroup STM32F769I_DISCOVERY_BSP_Public Functions BSP Public Functions
 * @brief This method returns the STM32F769I Discovery BSP Driver revision
 * @retval version: 0xXYZR (8bits for each
```c
uint32_t BSP_GetVersion(void)
{
    return __STM32F769I_DISCOVERY_BSP_VERSION;
}

/**
 * @brief Configures LED GPIO.
 * @param Led: LED to be configured.
 * This parameter can be one of the following values:
 * @arg LED1
 * @arg LED2
 * @retval None
 */
void BSP_LED_Init(Led_TypeDef Led)
{
    GPIO_InitTypeDef gpio_init_structure;

    LEDx_GPIO_CLK_ENABLE();
    /* Configure the GPIO_LED pin */
    gpio_init_structure.Pin    = GPIO_PIN[Led];
    gpio_init_structure.Mode   = GPIO_MODE_OUTPUT_PP;
    gpio_init_structure.Pull   = GPIO_PULLUP;
    gpio_init_structure.Speed  = GPIO_SPEED_HIGH;
    HAL_GPIO_Init(GPIO_PORT[Led], &gpio_init_structure);
}

/**
 * @brief DeInit LEDs.
 */
```
@param Led: LED to be configured.
This parameter can be one of the following values:

@arg LED1
@arg LED2

@note Led DeInit does not disable the GPIO clock

@retval None

```c
void BSP_LED_DeInit(Led_TypeDef Led)
{
    GPIO_InitTypeDef gpio_init_structure;

    /* DeInit the GPIO_LED pin */
    gpio_init_structure.Pin = GPIO_PIN[Led];

    /* Turn off LED */
    HAL_GPIO_WritePin(GPIO_PORT[Led], GPIO_PIN[Led], GPIO_PIN_RESET);
    HAL_GPIO_DeInit(GPIO_PORT[Led], gpio_init_structure.Pin);
}
```

@brief Turns selected LED On.
@param Led: LED to be set on
This parameter can be one of the following values:

@arg LED1
@arg LED2

@retval None

```c
void BSP_LED_On(Led_TypeDef Led)
{
    HAL_GPIO_WritePin(GPIO_PORT[Led], GPIO_PIN[Led], GPIO_PIN_SET);
}
```
/**
 * @brief Turns selected LED Off.
 * @param Led: LED to be set off
 * This parameter can be one of the following values:
 * @arg LED1
 * @arg LED2
 * @retval None
 */
void BSP_LED_Off(Led_TypeDef Led)
{
    HAL_GPIO_WritePin(GPIO_PORT[Led], GPIO_PIN[Led], GPIO_PIN_RESET);
}

/**
 * @brief Toggles the selected LED.
 * @param Led: LED to be toggled
 * This parameter can be one of the following values:
 * @arg LED1
 * @arg LED2
 * @retval None
 */
void BSP_LED_Toggle(Led_TypeDef Led)
{
    HAL_GPIO_TogglePin(GPIO_PORT[Led], GPIO_PIN[Led]);
}

/**
 * @brief Configures button GPIO and EXTI Line.
 * @param Button: Button to be configured
 * This parameter can be one of the following values:
 *
@arg BUTTON_WAKEUP: Wakeup Push Button
@arg BUTTON_USER: User Push Button

@param Button_Mode: Button mode

This parameter can be one of the following values:

@arg BUTTON_MODE_GPIO: Button will be used as simple IO
@arg BUTTON_MODE_EXTI: Button will be connected to EXTI line

@retval None

*/

void BSP_PB_Init(Button_TypeDef Button, ButtonMode_TypeDef Button_Mode)
{
    GPIO_InitTypeDef gpio_init_structure;
    BUTTON_GPIO_CLK_ENABLE();

    if(Button_Mode == BUTTON_MODE_GPIO)
    {
        /* Configure Button pin as input */
        gpio_init_structure.Pin = BUTTON_PIN[Button];
        gpio_init_structure.Mode = GPIO_MODE_INPUT;
        gpio_init_structure.Pull = GPIO_NOPULL;
        gpio_init_structure.Speed = GPIO_SPEED_FAST;
        HAL_GPIO_Init(BUTTON_PORT[Button], &gpio_init_structure);
    }
}
if(Button_Mode == BUTTON_MODE_EXTI) {
  /* Configure Button pin as input with External interrupt */
  gpio_init_structure.Pin = BUTTON_PIN[Button];
  gpio_init_structure.Pull = GPIO_NOPULL;
  gpio_init_structure.Speed = GPIO_SPEED_FAST;
  gpio_init_structure.Mode = GPIO_MODE_IT_RISING;
  HAL_GPIO_Init(BUTTON_PORT[Button], &gpio_init_structure);

  /* Enable and set Button EXTI Interrupt to the lowest priority */
  HAL_NVIC_SetPriority((IRQn_Type)(BUTTON_IRQn[Button]), 0x0F, 0x00);
  HAL_NVIC_EnableIRQ((IRQn_Type)(BUTTON_IRQn[Button]));
}

/**
 * @brief Push Button DeInit.
 * @param Button: Button to be configured
 * @note PB DeInit does not disable the GPIO clock
 * @retval None
 */
void BSP_PB_DeInit(Button_TypeDef Button) {
    GPIO_InitTypeDef gpio_init_structure;
    gpio_init_structure.Pin = BUTTON_PIN[Button];
    HAL_NVIC_DisableIRQ((IRQn_Type)(BUTTONIRQn[Button]));
    HAL_GPIO_DeInit(BUTTON_PORT[Button], gpio_init_structure.Pin);
}

/**< *
 * @brief Returns the selected button state.
 * @param Button: Button to be checked
 * This parameter can be one of the following values:
 * @arg BUTTON_WAKEUP: Wakeup Push Button
 * @arg BUTTON_USER: User Push Button
 * @retval The Button GPIO pin value
 */
uint32_t BSP_PB_GetState(Button_TypeDef Button) {
    return HAL_GPIO_ReadPin(BUTTON_PORT[Button], BUTTON_PIN[Button]);
}
*/
/**	@defgroup STM32F769I_DISCOVERY_LOW_LEVEL_Private_Functions STM32F769I_DISCOVERY_LOW_LEVEL Private Functions
*/

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

/******************************* I2C Routine ***************************/

/**
 * @brief Initializes I2C MSP.
 * @param i2c_handler : I2C handler
 * @retval None
 */
static void I2Cx_MspInit(I2C_HandleTypeDef * i2c_handler)
{
    GPIO_InitTypeDef gpio_init_structure;

    if (i2c_handler == (I2C_HandleTypeDef*)(&hI2cAudioHandler))
    {
        /******** Configure the GPIOs ********/
        /* Enable GPIO clock */
        DISCOVERY_AUDIO_I2Cx_SCL_GPIO_CLK_ENABLE();
        DISCOVERY_AUDIO_I2Cx_SDA_GPIO_CLK_ENABLE();
        /* Configure I2C Tx as alternate function */
        gpio_init_structure.Pin = DISCOVERY_AUDIO_
I2Cx_SCL_PIN;
00362  gpio_init_structure.Mode = GPIO_MODE_AF_OD;
00363  gpio_init_structure.Pull = GPIO_NOPULL;
00364  gpio_init_structure.Speed = GPIO_SPEED_FAST;
00365  gpio_init_structure.Alternate = DISCOVERY_AUDIO_I2Cx_SCL_AF;
00366  HAL_GPIO_Init(DISCOVERY_AUDIO_I2Cx_SCL_GPIO_PORT, &gpio_init_structure);

00367
00368  /* Configure I2C Rx as alternate function */
00369  gpio_init_structure.Pin = DISCOVERY_AUDIO_I2Cx_SDA_PIN;
00370  gpio_init_structure.Alternate = DISCOVERY_AUDIO_I2Cx_SDA_AF;
00371  HAL_GPIO_Init(DISCOVERY_AUDIO_I2Cx_SDA_GPIO_PORT, &gpio_init_structure);
00372
00373  /*** Configure the I2C peripheral ***/
00374  /* Enable I2C clock */
00375  DISCOVERY_AUDIO_I2Cx_CLK_ENABLE();
00376
00377  /* Force the I2C peripheral clock reset */
00378  DISCOVERY_AUDIO_I2Cx_FORCE_RESET();
00379
00380  /* Release the I2C peripheral clock reset */
00381  DISCOVERY_AUDIO_I2Cx_RELEASE_RESET();
00382
00383  /* Enable and set I2C1 Interrupt to a lower priority */
00384  HAL_NVIC_SetPriority(DISCOVERY_AUDIO_I2Cx_EV_IRQn, 0x0F, 0);
00385  HAL_NVIC_EnableIRQ(DISCOVERY_AUDIO_I2Cx_EV_IRQn);
/* Enable and set I2C1 Interrupt to a lower priority */
HAL_NVIC_SetPriority(DISCOVERY_AUDIO_I2Cx_ER_IRQn, 0x0F, 0);
HAL_NVIC_EnableIRQ(DISCOVERY_AUDIO_I2Cx_ER_IRQn);

else
{
    /*** Configure the GPIOs ***/
    /* Enable GPIO clock */
    DISCOVERY_EXT_I2Cx_SCL_SDA_GPIO_CLK_ENABLE();

    /* Configure I2C Tx as alternate function */
    gpio_init_structure.Pin = DISCOVERY_EXT_I2Cx_SCL_PIN;
gpio_init_structure.Mode = GPIO_MODE_AF_OD;
gpio_init_structure.Pull = GPIO_NOPULL;
gpio_init_structure.Speed = GPIO_SPEED_FAST;
gpio_init_structure.Alternate = DISCOVERY_EXT_I2Cx_SCL_SDA_AF;
HAL_GPIO_Init(DISCOVERY_EXT_I2Cx_SCL_SDA_GPIO_PORT, &gpio_init_structure);

    /* Configure I2C Rx as alternate function */
    gpio_init_structure.Pin = DISCOVERY_EXT_I2Cx_SDA_PIN;
HAL_GPIO_Init(DISCOVERY_EXT_I2Cx_SCL_SDA_GPIO_PORT, &gpio_init_structure);
/***	Configure the I2C peripheral ***/
/*	Enable I2C clock */
DISCOVERY_EXT_I2Cx_CLK_ENABLE();
/*	Force the I2C peripheral clock reset */
DISCOVERY_EXT_I2Cx_FORCE_RESET();
/*	Release the I2C peripheral clock reset */
DISCOVERY_EXT_I2Cx_RELEASE_RESET();
/*	Enable and set I2C1 Interrupt to a lower priority */
HAL_NVIC_SetPriority(DISCOVERY_EXT_I2Cx_EV_IRQn, 0x0F, 0);
HAL_NVIC_EnableIRQ(DISCOVERY_EXT_I2Cx_EV_IRQn);
/*	Enable and set I2C1 Interrupt to a lower priority */
HAL_NVIC_SetPriority(DISCOVERY_EXT_I2Cx_ER_IRQn, 0x0F, 0);
HAL_NVIC_EnableIRQ(DISCOVERY_EXT_I2Cx_ER_IRQn);
}
*/
**
* @brief Initializes I2C HAL.
* @param i2c_handler : I2C handler
* @retval None
*/
static void I2Cx_Init(I2C_HandleTypeDef *i2c_handler)
{
if(HAL_I2C_GetState(i2c_handler) == HAL_I2C_STATE_RESET)
if (i2c_handler == (I2C_HandleTypeDef*)(&hI2cAudioHandler)) {
    /* Audio and LCD I2C configuration */
    i2c_handler->Instance = DISCOVERY_AUDIO_I2Cx;
} else {
    /* External, camera and Arduino connector I2C configuration */
    i2c_handler->Instance = DISCOVERY_EXT_I2Cx;
}

i2c_handler->Init.Timing = DISCOVERY_I2Cx_TIMING;
i2c_handler->Init.OwnAddress1 = 0;
i2c_handler->Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
i2c_handler->Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
i2c_handler->Init.OwnAddress2 = 0;
i2c_handler->Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
i2c_handler->Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;

/* Init the I2C */
I2Cx_MspInit(i2c_handler);
HAL_I2C_Init(i2c_handler);

/**
 * @brief Reads multiple data.
 * @param i2c_handler : I2C handler

static HAL_StatusTypeDef I2Cx_ReadMultiple(I2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t *Buffer, uint16_t Length) {
    HAL_StatusTypeDef status = HAL_OK;

    status = HAL_I2C_Mem_Read(i2c_handler, Addr, (uint16_t)Reg, MemAddress, Buffer, Length, 1000);

    /* Check the communication status */
    if (status != HAL_OK) {
        /* I2C error occurred */
        I2Cx_Error(i2c_handler, Addr);
    }

    return status;
}
static HAL_StatusTypeDef I2Cx_WriteMultiple(I2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t *Buffer, uint16_t Length)
{
    HAL_StatusTypeDef status = HAL_OK;

    status = HAL_I2C_Mem_Write(i2c_handler, Addr, (uint16_t)Reg, MemAddress, Buffer, Length, 1000);

    /* Check the communication status */
    if (status != HAL_OK)
    {
        /* Re-Initialize the I2C Bus */
        I2Cx_Error(i2c_handler, Addr);
    }

    return status;
}

static HAL_StatusTypeDef I2Cx_IsDeviceReady(I2C_HandleTypeDef *i2c_handler, uint8_t DevAddress, uint8_t Trials)
{
    /* Check if target device is ready for communication. */
    /* This function is used with Memory devices */
    /* i2c_handler : I2C handler */
    /* DevAddress: Target device address */
    /* Trials: Number of trials */
    /* HAL status */

    return HAL_OK;
}
I2C_HandleTypeDef *i2c_handler, uint16_t DevAddress, uint32_t Trials)  
00523 {  
00524    return (HAL_I2C_IsDeviceReady(i2c_handler, DevAddress, Trials, 1000));  
00525 }  
00526  
00527    /**  
00528    * @brief Manages error callback by re-initializing I2C.  
00529    * @param i2c_handler : I2C handler  
00530    * @param Addr: I2C Address  
00531    * @retval None  
00532    */  
00533 static void I2Cx_Error(I2C_HandleTypeDef *i2c_handler, uint8_t Addr)  
00534 {  
00535    /* De-initialize the I2C communication bus  
00536    */  
00537    HAL_I2C_DeInit(i2c_handler);  
00538    /* Re-Initialize the I2C communication bus  
00539    */  
00540    I2Cx_Init(i2c_handler);  
00541 }  
00542  
00543    /**  
00544    */  
00545  
00546    /**********************************************************************  
00547    **********************************************************************  
00548    LINK OPERATIONS  
00549    /**********************************************************************  
00550    LINK AUDI
/**
 * @brief Initializes Audio low level.
 */
void AUDIO_IO_Init(void)
{
    I2Cx_Init(&hI2cAudioHandler);
}

/**
 * @brief DeInitializes Audio low level.
 */
void AUDIO_IO_DeInit(void)
{
}

/**
 * @briefWrites a single data.
 * @param Addr: I2C address
 * @param Reg: Reg address
 * @param Value: Data to be written
 * @retval None
 */
void AUDIO_IO_Write(uint8_t Addr, uint16_t Reg, uint16_t Value)
{
    uint16_t tmp = Value;
    Value = ((uint16_t)(tmp >> 8) & 0x00FF);
    Value |= ((uint16_t)(tmp << 8)& 0xFF00);
    I2Cx_WriteMultiple(&hI2cAudioHandler, Addr, Reg, I2C_MEMADD_SIZE_16BIT,(uint8_t*)&Value, 2);
}
/*
@brief Reads a single data.
@param Addr: I2C address
@param Reg: Reg address
@return Data to be read
*/

uint16_t AUDIO_IO_Read(uint8_t Addr, uint16_t Reg) {
    uint16_t read_value = 0, tmp = 0;

    I2Cx_ReadMultiple(&hI2cAudioHandler, Addr, Reg, I2C_MEMADD_SIZE_16BIT, (uint8_t*)&read_value, 2);
    tmp = ((uint16_t)(read_value >> 8) & 0x00FF);
    tmp |= ((uint16_t)(read_value << 8) & 0xFF00);
    read_value = tmp;
    return read_value;
}

void AUDIO_IO_Delay(uint32_t Delay) {
    HAL_Delay(Delay);
}

/******************************** LINK I2C E */
/**
 * @brief Initializes peripherals used by the I2C EEPROM driver.
 */

void EEPROM_IO_Init(void)
{
  I2Cx_Init(&hI2cExtHandler);
}

/**
 * @brief Write data to I2C EEPROM driver in using DMA channel.
 *
 * @param DevAddress: Target device address
 * @param MemAddress: Internal memory address
 * @param pBuffer: Pointer to data buffer
 * @paramBufferSize: Amount of data to be sent
 *
 * @retval HAL status
 */

HAL_StatusTypeDef EEPROM_IO_WriteData(uint16_t DevAddress, uint16_t MemAddress, uint8_t* pBuffer, uint32_t BufferSize)
{
  return (I2Cx_WriteMultiple(&hI2cExtHandler, DevAddress, MemAddress, I2C_MEMADD_SIZE_16BIT, pBuffer, BufferSize));
}

/**
 * @brief Read data from I2C EEPROM driver in using DMA channel.
 *
 * @param DevAddress: Target device address
 */
HAL_StatusTypeDef EEPROM_IO_ReadData(uint16_t DevAddress, uint16_t MemAddress, uint8_t* pBuffer, uint32_t BufferSize) {
    return (I2Cx_ReadMultiple(&hI2cExtHandler, DevAddress, MemAddress, I2C_MEMADD_SIZE_16BIT, pBuffer, BufferSize));
}

HAL_StatusTypeDef EEPROM_IO_IsDeviceReady(uint16_t DevAddress, uint32_t Trials) {
    return (I2Cx_IsDeviceReady(&hI2cExtHandler, DevAddress, Trials));
}
void TS_IO_Init(void)
{
    I2Cx_Init(&hI2cAudioHandler);
}

void TS_IO_Write(uint8_t Addr, uint8_t Reg, uint8_t Value)
{
    I2Cx_WriteMultiple(&hI2cAudioHandler, Addr, (uint16_t)Reg, I2C_MEMADD_SIZE_8BIT, (uint8_t*)&Value, 1);
}

uint8_t TS_IO_Read(uint8_t Addr, uint8_t Reg)
{
    uint8_t read_value = 0;
    I2Cx_ReadMultiple(&hI2cAudioHandler, Addr, Reg, I2C_MEMADD_SIZE_8BIT, (uint8_t*)&read_value,
return read_value;

} /* @brief Reads multiple data with I2C communication channel from TouchScreen. */

/* @param Addr: I2C address */
/* @param Reg: Register address */
/* @param Buffer: Pointer to data buffer */
/* @param Length: Length of the data */
/* @retval Number of read data */

uint16_t TS_IO_ReadMultiple(uint8_t Addr, uint8_t Reg, uint8_t *Buffer, uint16_t Length) {
    return I2Cx_ReadMultiple(&hI2cAudioHandler, Addr, (uint16_t)Reg, I2C_MEMADD_SIZE_8BIT, Buffer, Length);
}

/* @brief Writes multiple data with I2C communication channel from MCU to TouchScreen. */
/* @param Addr: I2C address */
/* @param Reg: Register address */
/* @param Buffer: Pointer to data buffer */
/* @param Length: Length of the data */
/* @retval None */

void TS_IO_WriteMultiple(uint8_t Addr, uint8_t Reg, uint8_t *Buffer, uint16_t Length) {
    I2Cx_WriteMultiple(&hI2cAudioHandler, Addr
, (uint16_t)Reg, I2C_MEMADD_SIZE_8BIT, Buffer, Length);
00728 }
00729
00730 /**
00731   * @brief Delay function used in TouchScreen low level driver.
00732   * @param Delay: Delay in ms
00733   * @retval None
00734 */
00735 void TS_IO_Delay(uint32_t Delay)
00736 {  
00737    HAL_Delay(Delay);
00738 }
00739
00740 /*************************************************************** LINK OTM8009A (Display driver) *******************************************************************/
00741 /**
00742   * @brief OTM8009A delay
00743   * @param Delay: Delay in ms
00744 */
00745 void OTM8009A_IO_Delay(uint32_t Delay)
00746 {  
00747    HAL_Delay(Delay);
00748 }
00749
00750 /*************************************************************** LINK ADV7533 DSI-HDMI (Display driver) ********************************************/ 
00751 /**
00752   * @brief Initializes HDMI IO low level.
00753   * @retval None
00754 */
00755 void HDMI_IO_Init(void)
00756 {  
00757    I2Cx_Init(&I2cAudioHandler);
00758 }
00759
/**
 * @brief HDMI writes single data.
 * @param Addr: I2C address
 * @param Reg: Register address
 * @param Value: Data to be written
 * @retval None
 */

void HDMI_IO_Write(uint8_t Addr, uint8_t Reg, uint8_t Value)
{
    I2Cx_WriteMultiple(&hI2cAudioHandler, Addr, (uint16_t)Reg, I2C_MEMADD_SIZE_8BIT, &Value, 1);
}

/**
 * @brief Reads single data with I2C communication channel from HDMI bridge.
 * @param Addr: I2C address
 * @param Reg: Register address
 * @retval Read data
 */

uint8_t HDMI_IO_Read(uint8_t Addr, uint8_t Reg)
{
    uint8_t value = 0x00;
    I2Cx_ReadMultiple(&hI2cAudioHandler, Addr, (uint16_t)Reg, I2C_MEMADD_SIZE_8BIT, &value, 1);
    return value;
}

/**
 * @brief HDMI delay
 * @param Delay: Delay in ms

```c
00791    * @retval None
00792    */
00793    void HDMI_IO_Delay(uint32_t Delay)
00794    {
00795        HAL_Delay(Delay);
00796    }
00797    /**
00798    * @}
00799    */
00800    /**
00801    * @}
00802    */
00803    */
00804    /**
00805    * @}
00806    */
00807    */
00808    /**
00809    * @}
00810    */
00811    */
00812    /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/```
/**
 ** ******************************************************************************
 ** ******************************************************************************
 **
 ** @file      stm32f769i_discovery_audio.h
 ** @author    MCD Application Team
 ** @version   V2.0.0
 ** @date      30-December-2016
 ** @brief     This file contains the common defines and functions prototypes for
 **             the stm32f769i_discovery_audio.c driver.
 **
 ** @attention
 **
 ** &lt;h2&gt;&lt;center&gt;&lt;copy;&lt;COPYRIGHT(c) 2016 STM Microelectronics&lt;/center&gt;&lt;/h2&gt;
 **
 ** Redistribution and use in source and binary forms, with or without modification,
 ** are permitted provided that the following conditions are met:
 **
 ** ******************************************************************************
 ** ******************************************************************************
 **
 ** &lt;h2&gt;&lt;center&gt;&lt;copy;&lt;COPYRIGHT(c) 2016 STM Microelectronics&lt;/center&gt;&lt;/h2&gt;
 **
1. Redistributions of source code must retain the above copyright notice,
   this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.
3. Neither the name of STMicroelectronics nor the names of its contributors
   may be used to endorse or promote products derived from this software
   without specific prior written permission.

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

#ifndef __STM32F769I_DISCOVERY_AUDIO_H
#define __STM32F769I_DISCOVERY_AUDIO_H

#ifdef __cplusplus
extern "C"
{

/* Includes ----------------------------------------------*/

/* Include audio component Driver */
#include "../Components/wm8994/wm8994.h"
#include "stm32f769i_discovery.h"
#include <stdlib.h>

/** @addtogroup BSP
 * @{
 */

/** @addtogroup STM32F769I_DISCOVERY
 * @{
 */

/** @defgroup STM32F769I_DISCOVERY_AUDIO STM32F769I_DISCOVERY_AUDIO
 * @{
 */

/** @defgroup STM32F769I_DISCOVERY_AUDIO_Exported_Types STM32F769I_DISCOVERY_AUDIO_Exported_Types
 * @{
 */

/** @}*/

/** @}*/

/** @}*/

```
/**	@defgroup STM32F769I_DISCOVERY_AUDIO_Exp Ported_Constants STM32F769I_DISCOVERY_AUDIO_Exported_Constants
*/

/**	@defgroup BSP_Audio_Sample_Rate BSP_Audio_Sample_Rate
*/

#define BSP_AUDIO_FREQUENCY_96K SAI_AUDIO_FREQUENCY_96K
#define BSP_AUDIO_FREQUENCY_48K SAI_AUDIO_FREQUENCY_48K
#define BSP_AUDIO_FREQUENCY_44K SAI_AUDIO_FREQUENCY_44K
#define BSP_AUDIO_FREQUENCY_32K SAI_AUDIO_FREQUENCY_32K
#define BSP_AUDIO_FREQUENCY_22K SAI_AUDIO_FREQUENCY_22K
#define BSP_AUDIO_FREQUENCY_16K SAI_AUDIO_FREQUENCY_16K
#define BSP_AUDIO_FREQUENCY_11K SAI_AUDIO_FREQUENCY_11K
#define BSP_AUDIO_FREQUENCY_8K SAI_AUDIO_FREQUENCY_8K

/**	@}
*/
USER SAI defines parameters

/** CODEC_AudioFrame_SLOT_TDMMode In W8994 codec the Audio frame contains 4 slots: TDM Mode */

* TDM format:
* +------------------|------------------|-
| CODEC_SLOT0 Left | CODEC_SLOT1 Left |
| CODEC_SLOT0 Right | CODEC_SLOT1 Right |
* +---------------------------------------+

/* To have 2 separate audio stream in Both headphone and speaker the 4 slot must be activated */
#define CODEC_AUDIOFRAME_SLOT_0123
  SAI_SLOTACTIVE_0 | SAI_SLOTACTIVE_1 | SAI_SLOTACTIVE_2 | SAI SLOTACTIVE_3

/* To have an audio stream in headphone only SAI Slot 0 and Slot 2 must be activated */
#define CODEC_AUDIOFRAME_SLOT_02
  SAI SLOTACTIVE_0 | SAI SLOTACTIVE_2

/* To have an audio stream in speaker only SAI Slot 1 and Slot 3 must be activated */
#define CODEC_AUDIOFRAME_SLOT_13
  SAI SLOTACTIVE_1 | SAI SLOTACTIVE_3

/* SAI peripheral configuration defines */
#define AUDIO_OUT_SAIx
  SAI1_Block_A

#define AUDIO_OUT_SAIx_CLK_ENABLE()
__HAL_RCC_SAI1_CLK_ENABLE()

#define AUDIO_OUT_SAIx_CLK_DISABLE()
__HAL_RCC_SAI1_CLK_DISABLE()

#define AUDIO_OUT_SAIx_AF
GPIO_AF6_SAI1

#define AUDIO_OUT_SAIx_MCLK_ENABLE()
__HAL_RCC_GPIOG_CLK_ENABLE()

#define AUDIO_OUT_SAIx_MCLK_GPIO_PORT
GPIOG

#define AUDIO_OUT_SAIx_MCLK_PIN
GPIO_PIN_7

#define AUDIO_OUT_SAIx_SD_FS_CLK_ENABLE()
__HAL_RCC_GPIOE_CLK_ENABLE()

#define AUDIO_OUT_SAIx_SD_FS_SCK_GPIO_PORT
GPIOE

#define AUDIO_OUT_SAIx_SD_FS_SCK_GPIO_PIN
GPIO_PIN_4

#define AUDIO_OUT_SAIx_FS_PIN
GPIO_PIN_5

#define AUDIO_OUT_SAIx_SD_PIN
GPIO_PIN_6

/* SAI DMA Stream definitions */
#define AUDIO_OUT_SAIx_DMAx_CLK_ENABLE()
__HAL_RCC_DMA2_CLK_ENABLE()

#define AUDIO_OUT_SAIx_DMAx_STREAM
DMA2_Stream1

#define AUDIO_OUT_SAIx_DMAx_CHANNEL
DMA_CHANNEL_0

#define AUDIO_OUT_SAIx_DMAx_IRQ
DMA2_Stream1_IRQn

#define AUDIO_OUT_SAIx_DMAx_PERIPH_DATA_SIZE
DMA_PDATAALIGN_HALFWORD

#define AUDIO_OUT_SAIx_DMAx_MEM_DATA_SIZE
DMA_MDATAALIGN_HALFWORD

#define DMA_MAX_SZE
0xFFFF
00132 00133 #define AUDIO_OUT_SAIx_DMAx_IRQHandler
            DMA2_Stream1_IRQHandler
00134 00135 /* Select the interrupt preemption priority
            and subpriority for the DMA interrupt */
00136 00137 00138 #define AUDIO_OUT_IRQ_PREPRIO
            ((uint32_t)0x0E)
00139 00140 /*------------------------------------------
            ------------------------------------
            AUDIO IN CONFIGURATION
            --------------------------------------------
            ----------------------------------*/
00141 /* SAI peripheral configuration defines */
00142 00143 #define AUDIO_IN_SAIx
            SAI1_Block_B
00144 00145 #define AUDIO_IN_SAIx_CLK_ENABLE()
            __HAL_RCC_SAI1_CLK_ENABLE()
00146 00147 #define AUDIO_IN_SAIx_CLK_DISABLE()
            __HAL_RCC_SAI1_CLK_DISABLE()
00148 00149 #define AUDIO_IN_SAIx_AF
            GPIO_AF6_SAI1
00150 00151 #define AUDIO_IN_SAIx_SD_ENABLE()
            __HAL_RCC_GPIOE_CLK_ENABLE()
00152 00153 #define AUDIO_IN_SAIx_SD_GPIO_PORT
            GPIOE
00154 00155 #define AUDIO_IN_SAIx_SD_PIN
            GPIO_PIN_3
00156 00157 /* SAI DMA Stream definitions */
00158 00159 #define AUDIO_IN_SAIx_DMAx_CLK_ENABLE()
            __HAL_RCC_DMA2_CLK_ENABLE()
00160 00161 #define AUDIO_IN_SAIx_DMAx_STREAM
            DMA2_Stream4
#define AUDIO_IN_SAIx_DMAx_CHANNEL DMA_CHANNEL_1
#define AUDIO_IN_SAIx_DMAx_IRQ DMA2_Stream4_IRQn
#define AUDIO_IN_SAIx_DMAx_PERIPH_DATA_SIZE DMA_PDATAALIGN_HALFWORD
#define AUDIO_IN_SAIx_DMAx_MEM_DATA_SIZE DMA_MDATAALIGN_HALFWORD

#define AUDIO_IN_INT_GPIO_ENABLE() __HAL_RCC_GPIOJ_CLK_ENABLE()
#define AUDIO_IN_INT_GPIO_PORT GPIOJ
#define AUDIO_IN_INT_GPIO_PIN GPIO_PIN_12
#define AUDIO_IN_INT_IRQ EXTI15_10_IRQn

/* DFSDM Configuration defines */
#define AUDIO_DFSDMx_TOP_RIGHT_CHANNEL DFSDM_CHANNEL_0
#define AUDIO_DFSDMx_TOP_LEFT_CHANNEL DFSDM_CHANNEL_1
#define AUDIO_DFSDMx_BUTTOM_RIGHT_CHANNEL DFSDM_CHANNEL_4
#define AUDIO_DFSDMx_BUTTOM_LEFT_CHANNEL DFSDM_CHANNEL_5

#define AUDIO_DFSDMx_TOP_LEFT_FILTER DFSDM1_Filter0
#define AUDIO_DFSDMx_TOP_RIGHT_FILTER DFSDM1_Filter1
#define AUDIO_DFSDMx_BUTTOM_LEFT_FILTER DFSDM1_Filter2
#define AUDIO_DFSDMx_BUTTOM_RIGHT_FILTER DFSDM1_Filter3
/* DFSDM DMA Right and Left channels definitions */
#define AUDIO_DFSDMx_DMAx_CLK_ENABLE() __HAL_RCC_DMA2_CLK_ENABLE()
#define AUDIO_DFSDMx_DMAx_CHANNEL DMA_CHANNEL_8
#define AUDIO_DFSDMx_DMAx_PERIPH_DATA_SIZE DMA_PDATAALIGN_WORD
#define AUDIO_DFSDMx_DMAx_MEM_DATA_SIZE DMA_MDATAALIGN_WORD
#define AUDIO_DFSDMx_DMAx_TOP_LEFT_STREAM DMA2_Stream0
#define AUDIO_DFSDMx_DMAx_TOP_LEFT_IRQ DMA2_Stream0_IRQn
#define AUDIO_DFSDMx_DMAx_TOP_LEFT_IRQHandle
/** Select the interrupt preemption priority and subpriority for the DMA interrupt */
#define AUDIO_IN_IRQ_PREPRIO ((uint32_t)0x0F)

CONFIGURATION: Audio Driver Configuration parameters
#define AUDIODATA_SIZE
/* 16-bits audio data size */

#define AUDIO_OK ((uint8_t)0)
#define AUDIO_ERROR ((uint8_t)1)
#define AUDIO_TIMEOUT ((uint8_t)2)

/* Audio In default settings */
#define DEFAULT_AUDIO_IN_FREQ BSP_AUDIO_FREQUENCY_16K
#define DEFAULT_AUDIO_IN_BIT_RESOLUTION ((uint8_t)16)
#define DEFAULT_AUDIO_IN_CHANNEL_NBR ((uint8_t)2)
#define DEFAULT_AUDIO_IN_VOLUME ((uint16_t)64)

/*------------------------------------------
-----------------------------------------
OUTPUT DEVICES definition
-----------------------------------------
------------------------------------------*/

#define OUTPUT_DEVICE_HEADPHONE1 OUTPUT_DEVICE_HEADPHONE
#define OUTPUT_DEVICE_HEADPHONE2 OUTPUT_DEVICE_SPEAKER
/* Headphone2 is connected to Speaker output of the wm8994 */

/*------------------------------------------
-----------------------------------------
INPUT DEVICES definition
-----------------------------------------
------------------------------------------*/
/**
 * @defgroup STM32F769I_DISCOVERY_AUDIO_Exported_Macros
 * STM32F769I_DISCOVERY_AUDIO Exported Macros
 *
 */

#define INPUT_DEVICE_DIGITAL_MIC ((uint16_t)0)

/*	Analog microphone input from 3.5 audio jack connector */
#define INPUT_DEVICE_ANALOG_MIC INPUT_DEVICE_INPUT_LINE_1

/**
 * @defgroup STM32F769I_DISCOVERY_AUDIO_Exported_Functions
 * STM32F769I_DISCOVERY_AUDIO_Exported Functions
 *
 */

uint8_t BSP_AUDIO_OUT_Init(uint16_t OutputDevice, uint8_t Volume, uint32_t AudioFreq);
void BSP_AUDIO_OUT_DeInit(void);
uint8_t BSP_AUDIO_OUT_Play(uint16_t* pBuffer, uint32_t Size);
void BSP_AUDIO_OUT_ChangeBuffer(uint16_t *pData, uint16_t Size);
uint8_t BSP_AUDIO_OUT_Pause(void);
uint8_t BSP_AUDIO_OUT_Resume(void);
uint8_t BSP_AUDIO_OUT_Stop(uint32_t Option);
uint8_t BSP_AUDIO_OUT_SetVolume(uint8_t Volume);
void BSP_AUDIO_OUT_SetFrequency(uint32_t AudioFreq);
void BSP_AUDIO_OUT_SetAudioFrameSlot(uint32_t AudioFrameSlot);
uint8_t BSP_AUDIO_OUT_SetMute(uint32_t Cmd);
uint8_t BSP_AUDIO_OUT_SetOutputMode(uint8_t Output);

/* User Callbacks: user has to implement these functions in his code if they are needed. */
void BSP_AUDIO_OUT_TransferComplete_CallBack(void);

/* This function is called when half of the requested buffer has been transferred. */
void BSP_AUDIO_OUT_HalfTransfer_CallBack(void);

/* This function is called when an Interrupt due to transfer error on or peripheral error occurs. */
void BSP_AUDIO_OUT_Error_CallBack(void);

/* These functions can be modified in case the current settings (e.g. DMA stream) need to be changed for specific application needs */
void BSP_AUDIO_OUT_ClockConfig(SAI_HandleTy
peDef *hsai, uint32_t AudioFreq, void *Params);
00285 void BSP_AUDIO_OUT_MspInit(SAI_HandleTypeDef *hsai, void *Params);
00286 void BSP_AUDIO_OUT_MspDeInit(SAI_HandleTypeDef *hsai, void *Params);
00287
00288 /**
00289 * @}
00290 */
00291
00292 /** @defgroup STM32F769I_DISCOVERY_AUDIO_IN_ Exported_Functions STM32F769I_DISCOVERY_AUDIO_IN E xported Functions
00293 * @{
00294 */
00295 uint8_t BSP_AUDIO_IN_Init(uint32_t AudioFreq,
00296 uint32_t BitRes, uint32_t ChnlNbr);
00297 uint8_t BSP_AUDIO_IN_InitEx(uint16_t InputDevice, uint32_t AudioFreq, uint32_t BitRes, uint32_t ChnlNbr);
00298 uint8_t BSP_AUDIO_IN_AllocScratch (int32_t *pScratch, uint32_t size);
00299 uint8_t BSP_AUDIO_IN_GetChannelNumber(void);
00300 void BSP_AUDIO_IN_DeInit(void);
00301 uint8_t BSP_AUDIO_IN_Record(uint16_t *pData, uint32_t Size);
00302 uint8_t BSP_AUDIO_IN_Stop(void);
00303 uint8_t BSP_AUDIO_IN_Pause(void);
00304 uint8_t BSP_AUDIO_IN_Resume(void);
00305 /* User Callbacks: user has to implement these functions in his code if they are needed. */
00306 /* This function should be implemented by the user application.**/
00307 It is called into this driver when the current buffer is filled to prepare the next
00308 buffer pointer and its size. */
void BSP_AUDIO_IN_TransferComplete_CallBack(void);
void BSP_AUDIO_IN_HalfTransfer_CallBack(void);

/*
 * This function is called when an Interrupt due to transfer error on or peripheral error occurs. */
void BSP_AUDIO_IN_Error_CallBack(void);

/*
 * These function can be modified in case the current settings (e.g. DMA stream) need to be changed for specific application needs */
void BSP_AUDIO_IN_ClockConfig(DFSDM_Filter_HandleTypeDef *hdfsdm_filter, uint32_t AudioFreq, void *Params);
void BSP_AUDIO_IN_MspInit(void);
void BSP_AUDIO_IN_MspDeInit(void);

/** *
 * @}
 */

/** *
 * @}
 */

/** *
 * @}
 */

/** *
 * @}
 */

#define __cplusplus
} #endif
#endif /* __STM32F769I_DISCOVERY_AUDIO_H */
/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE*****/
stm32f769i_discovery_eeprom.h

Go to the documentation of this file.

```c
/**
 * @file    stm32f769i_discovery_eeprom.h
 * @author  MCD Application Team
 * @version V2.0.0
 * @date    30-December-2016
 * @brief   This file contains all the functions prototypes for
 *          the stm32f769i_discovery_eeprom.c firmware driver.
 */

```

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

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
/* Define to prevent recursive inclusion ----------------------------------*/
#ifndef __STM32F769I_DISCOVERY_EEPROM_H
#define __STM32F769I_DISCOVERY_EEPROM_H

#ifdef __cplusplus
extern "C"
#else
#endif

/* Includes --------------------------------------------------------------*/
#include "stm32f769i_discovery.h"

/** @addtogroup BSP
 * @{
 * /

/** @addtogroup STM32F769I_DISCOVERY
 * @{
 * /

/** @addtogroup STM32F769I_DISCOVERY_EEPROM
 * @brief This file includes the I2C EEPROM driver of STM32F769I-DISCOVERY board.
 * @{
 * /

/** @addtogroup STM32F769I_DISCOVERY_EEPROM_Ex
ported_Types EEPROM Exported Types
 * @{
 * /

/** */
* @}
*/

/** @defgroup STM32F769I_DISCOVERY_EEPROM_Exported_Constants EEPROM Exported Constants */

/* EEPROM hardware address and page size */
#define EEPROM_PAGESIZE ((uint8_t)4)
#define EEPROM_MAX_SIZE ((uint16_t)0x2000) /* 64Kbit */

/* Maximum number of trials for EEPROM_WaitEepromStandbyState() function */
#define EEPROM_MAX_TRIALS ((uint32_t)3000)

#define EEPROM_OK ((uint32_t)0)
#define EEPROM_FAIL ((uint32_t)1)
#define EEPROM_TIMEOUT ((uint32_t)2)

/** @defgroup STM32F769I_DISCOVERY_EEPROM_Exported_Macros EEPROM Exported Macros */

* @{
*/

/** @defgroup STM32F769I_DISCOVERY_EEPROM_Exported_Macros EEPROM Exported Macros */

* @{
*/

* @}
/** @defgroup STM32F769I_DISCOVERY_EEPROM_Exported_Functions EEPROM Exported Functions */

uint32_t BSP_EEPROM_Init(void);
uint8_t  BSP_EEPROM_DeInit(void);
uint32_t BSP_EEPROM_ReadBuffer(uint8_t* pBuffer, uint16_t ReadAddr, uint16_t* NumByteToRead);
uint32_t BSP_EEPROM_WritePage(uint8_t* pBuffer, uint16_t WriteAddr, uint8_t* NumByteToWrite);
uint32_t BSP_EEPROM_WriteBuffer(uint8_t* pBuffer, uint16_t WriteAddr, uint16_t NumByteToWrite);
uint32_t BSP_EEPROM_WaitEepromStandbyState(void);

/* USER Callbacks: This function is declared as __weak in EEPROM driver and should be implemented into user application.
BSP_EEPROM_TIMEOUT_UserCallback() function is called whenever a timeout condition occurs during communication (waiting on an event that doesn't occur, bus errors, busy devices ...). */
void BSP_EEPROM_TIMEOUT_UserCallback(void);

/* Link function for I2C EEPROM peripheral */

void EEPROM_IO_Init(void);
HAL_StatusTypeDef EEPROM_IO_WriteData(uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize);
HAL_StatusTypeDef EEPROM_IO_ReadData(uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize);
HAL_StatusTypeDef EEPROM_IO_IsDeviceReady(uint16_t DevAddress, uint32_t Trials);

/**
 * @}
 */

/**
 * @}
 */

/**
 * @}
 */

/**
 * @}
 */

#ifdef __cplusplus
} /*__STM32F769I_DISCOVERY_EEPROM_H */
#endif /*__cplusplus*/

/**
 * __STM32F769I_DISCOVERY_EEPROM_H */

/**
 * (C) COPYRIGHT STMicroelectronics ****END OF FILE****/

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery
BSP User Manual by doxygen 1.7.6.1
stm32f769i_discovery_eeprom.c

Go to the documentation of this file.

```c
/**
 * @file stm32f769i_discovery_eeprom.c
 * @author MCD Application Team
 * @version V2.0.0
 * @date 30-December-2016
 * @brief This file provides a set of functions needed to manage an I2C M24LR64 EEPROM memory.

*verbatim
To be able to use this driver, the switch EE_M24LR64 must be defined in your toolchain compiler preprocessor

Notes:
- The I2C EEPROM memory (M24LR64) is available on separate daughter board ANT7-M24LR-A, which is
```
not provided with the STM32F769I_DISCOVERY board.

To use this driver you have to connect the ANT7-M24LR-A to CN2 connector of STM32F769I_DISCOVERY board.

It implements a high level communication layer for read and write from/to this memory. The needed STM32F7xx hardware resources (I2C and GPIO) are defined in stm32f769i_discovery.h file, and the initialization is performed in EEPROM_IO_Init() function declared in stm32f769i_discovery.c file.

You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and EEPROM_IO_Init() function.

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

@note Regarding BSP_EEPROM_WritePage(), it is an optimized function to perform small write (less than 1 page) BUT the number of bytes (combined to write start address) must not cross the EEPROM page bound
This function can only writes into the boundaries of an EEPROM page.

This function doesn't check on boundaries condition (in this driver the function BSP_EEPROM_WriteBuffer() which calls BSP_EEPROM_WritePage() is responsible of checking on Page boundaries).

+-----------------------------------------------------------------+
| Pin assignment for M24LR64 EEPROM |
+-----------------------------------------------------------------+
| | STM32F7xx I2C Pins |
| | EEPROM | Pin |
+------------------------------------+-----------------------------+
| . | E0(GND) | 1 (0V) |
| . | AC0 | 2 |
| . | AC1 | 3 |
| . | VSS | 4 (0V) |
| | SDA | 5 |
| | SDA | 5 |
| | SCL | 6 |
| . | E1(GND) | 7 (0V) |
| . | VDD | 8 (3.3V) |
* @attention

<h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>

* Redistribution and use in source and binary forms, with or without modification,
* are permitted provided that the following conditions are met:
* 1. Redistributions of source code must retain the above copyright notice,
* this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright notice,
* this list of conditions and the following disclaimer in the documentation
* and/or other materials provided with the distribution.
* 3. Neither the name of STMicroelectronics nor the names of its contributors
* may be used to endorse or promote products derived from this software
* without specific prior written permission.
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
* DISCLAIMER. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
* SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
* CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
* OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

#include "stm32f769i_discovery_eeprom.h"

/** @addtogroup BSP
 */
/** @addtogroup STM32F769I_DISCOVERY
 */
/** @addtogroup STM32F769I_DISCOVERY_EEPROM
 STM32F769I_DISCOVERY EEPROM
 */
/** @brief This file includes the I2C EEPROM driver of STM32F769I-DISCOVERY board.
 */
/**
 * @defgroup STM32F769I_DISCOVERY_EEPROM_Private_Types EEPROM Private Types
 */

/**
 * @defgroup STM32F769I_DISCOVERY_EEPROM_Private_Defines EEPROM Private Defines
 */

/**
 * @defgroup STM32F769I_DISCOVERY_EEPROM_Private_Macros EEPROM Private Macros
 */

/**
 * @defgroup STM32F769I_DISCOVERY_EEPROM_Private_Variables EEPROM Private Variables
 */

/**
 * @defgroup STM32F769I_DISCOVERY_EEPROM_Private_Function_Prototypes EEPROM Private Function Prototypes
 */

__IO uint16_t EEPROMAddress = 0;
__IO uint16_t EEPROMDataRead;
__IO uint8_t EEPROMDataWrite;
/**	@defgroup STM32F769I_DISCOVERY_EEPROM_Private_Functions EEPROM Private Functions
/**	@{
    *	@brief 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)
    *	@retval EEPROM_OK (0) if operation is correctly performed, else return value different from EEPROM_OK (0)
 */

uint32_t BSP_EEPROM_Init(void)
{
    /* I2C Initialization */
    EEPROM_IO_Init();

    /* Select the EEPROM address for A01 and check if OK */
    EEPROMAddress = EEPROM_I2C_ADDRESS_A01;
    if(EEPROM_IO_IsDeviceReady(EEPROMAddress, EEPROM_MAX_TRIALS) != HAL_OK)
    {
        /* Select the EEPROM address for A02 and
check if OK */

EEPROMAddress = EEPROM_I2C_ADDRESS_A02;
if (EEPROM_IO_IsDeviceReady(EEPROMAddress, EEPROM_MAX_TRIALS) != HAL_OK)
{
    return EEPROM_FAIL;
}
return EEPROM_OK;

/**
 * @brief DeInitializes the EEPROM.
 * @retval EEPROM state
 */
uint8_t BSP_EEPROM_DeInit(void)
{
/* I2C won't be disabled because common to other functionalities */
return EEPROM_OK;
}

/**
 * @brief Reads a block of data from the EEPROM.
 * @param pBuffer: pointer to the buffer that receives the data read from.
 * @param ReadAddr: EEPROM's internal address to start reading from.
 * @param 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 EEPR
OM. Application should monitor this variable in order know when the transfer is complete.

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

 */

uint32_t BSP_EEPROM_ReadBuffer(uint8_t* pBuffer, uint16_t ReadAddr, uint16_t* NumByteToRead)
{
  uint32_t buffersize = *NumByteToRead;

  /* Set the pointer to the Number of data to be read. This pointer will be used by the DMA Transfer Completer interrupt Handler in order to reset the variable to 0. User should check on this variable in order to know if the DMA transfer has been complete or not. */

  EEPROMDataRead = *NumByteToRead;

  if(EEPROM_IO_ReadData(EEPROMAddress, ReadAddr, pBuffer, buffersize) != HAL_OK)
  {
    BSP_EEPROM_TIMEOUT_UserCallback();
    return EEPROM_FAIL;
  }

  /* If all operations OK, return EEPROM_OK (0) */
  return EEPROM_OK;
}
* @brief Writes more than one byte to the EEPROM with a single WRITE cycle.
* @note The number of bytes (combined to write start address) must not cross the EEPROM page boundary. This function can only write into the boundaries of an EEPROM page.

* This function doesn't check on boundaries condition (in this driver the function BSP_EEPROM_WriteBuffer() which calls BSP_EEPROM_WritePage() is responsible of checking on Page boundaries).

* @param pBuffer: pointer to the buffer containing the data to be written to the EEPROM.

* @param WriteAddr: EEPROM's internal address to write to.

* @param 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.

* @note This function just configure the communication and enable the DMA channel to transfer data. Meanwhile, the user application may perform other tasks in parallel.
*@retval EEPROM_OK (0) if operation is correctly performed, else return value different from EEPROM_OK (0) or the timeout user callback.*

```
uint32_t BSP_EEPROM_WritePage(uint8_t* pBuffer, uint16_t WriteAddr, uint8_t* NumByteToWrite)
{
    uint32_t buffersize = *NumByteToWrite;
    uint32_t status = EEPROM_OK;

    /* Set the pointer to the Number of data to be written. This pointer will be used by the DMA Transfer Completer interrupt Handler in order to reset the variable to 0. User should check on this variable in order to know if the DMA transfer has been complete or not. */
    EEPROMDataWrite = *NumByteToWrite;

    if(EEPROM_IO_WriteData(EEPROMAddress, WriteAddr, pBuffer, buffersize) != HAL_OK)
    {
        BSP_EEPROM_TIMEOUT_UserCallback();
        status = EEPROM_FAIL;
    }

    if(BSP_EEPROM_WaitEepromStandbyState() != EEPROM_OK)
    {
        return EEPROM_FAIL;
    }

    /* If all operations OK, return EEPROM_OK (0) */
```
return status;
}
/**
 * @brief Writes buffer of data to the I2C EEPROM.
 * @param pBuffer: pointer to the buffer containing the data to be written to the EEPROM.
 * @param WriteAddr: EEPROM's internal address to write to.
 * @param NumByteToWrite: number of bytes to write to the EEPROM.
 * @retval EEPROM_OK (0) if operation is correctly performed, else return value different from EEPROM_OK (0) or the timeout user callback.
 */
uint32_t BSP_EEPROM_WriteBuffer(uint8_t *pBuffer, uint16_t WriteAddr, uint16_t NumByteToWrite) {
    uint16_t numofpage = 0, numofsingle = 0, count = 0;
    uint16_t addr = 0;
    uint8_t dataindex = 0;
    uint32_t status = EEPROM_OK;

    addr = WriteAddr % EEPROM_PAGESIZE;
    count = EEPROM_PAGESIZE - addr;
    numofpage = NumByteToWrite / EEPROM_PAGESIZE;
    numofsingle = NumByteToWrite % EEPROM_PAGESIZE;

    /* If WriteAddr is EEPROM_PAGESIZE aligned */
    if(addr == 0)
if (numofpage == 0) {
    /* Store the number of data to be written */
    dataindex = numofsingle;
    /* Start writing data */
    status = BSP_EEPROM_WritePage(pBuffer, WriteAddr, (uint8_t*)(&dataindex));
    if (status != EEPROM_OK) {
        return status;
    }
}
/* If NumByteToWrite > EEPROM_PAGESIZE */
else {
    while (numofpage--) {
        /* Store the number of data to be written */
        dataindex = EEPROM_PAGESIZE;
        status = BSP_EEPROM_WritePage(pBuffer, WriteAddr, (uint8_t*)(&dataindex));
        if (status != EEPROM_OK) {
            return status;
        }
        WriteAddr += EEPROM_PAGESIZE;
        pBuffer += EEPROM_PAGESIZE;
    }
    if (numofsingle!=0)
00325  
00326  /* Store the number of data to be written */
00327  dataindex = numofsingle;
00328  status = BSP_EEPROM_WritePage(pBuffer, WriteAddr, (uint8_t*)(&dataindex));
00329  if(status != EEPROM_OK)
00330      { 
00331          return status;
00332      }
00333  
00334  } 
00335 
00336  /* If WriteAddr is not EEPROM_PAGESIZE aligned */
00337  else 
00338      {
00339          /* If NumByteToWrite < EEPROM_PAGESIZE */
00340          
00341              
00342                  /* If the number of data to be written is more than the remaining space in the current page: */
00343              
00344                  
00345                      
00346                          /* Store the number of data to be written */
00347              
00348              /* Write the data contained in same page */
00349              status = BSP_EEPROM_WritePage(pBuffer, WriteAddr, (uint8_t*)(&dataindex));
00350          if(status != EEPROM_OK)
00351              
00352              return status;
00353      }
/* Store the number of data to be written */
dataindex = (NumByteToWrite - count);

/* Write the remaining data in the following page */
status = BSP_EEPROM_WritePage((uint8_t*)(pBuffer + count), (WriteAddr + count), (uint8_t*)(&dataindex));
if(status != EEPROM_OK)
 {
   return status;
 }
else
 {
   /* Store the number of data to be written */
dataindex = numofsingle;
status = BSP_EEPROM_WritePage(pBuffer, WriteAddr, (uint8_t*)(&dataindex));
if(status != EEPROM_OK)
 {
   return status;
 }

/* If NumByteToWrite > EEPROM_PAGESIZE */
else
 {
   NumByteToWrite -= count;
   numofpage = NumByteToWrite / EEPROM_PAGESIZE;
   numofsingle = NumByteToWrite % EEPROM_PAGESIZE;
if(count != 0) {
  /* Store the number of data to be written */
  dataindex = count;
  status = BSP_EEPROM_WritePage(pBuffer, WriteAddr, (uint8_t*)(&dataindex));
  if(status != EEPROM_OK) {
    return status;
  }
  WriteAddr += count;
  pBuffer += count;
}

while(numofpage--)
{
  /* Store the number of data to be written */
  dataindex = EEPROM_PAGESIZE;
  status = BSP_EEPROM_WritePage(pBuffer, WriteAddr, (uint8_t*)(&dataindex));
  if(status != EEPROM_OK) {
    return status;
  }
  WriteAddr += EEPROM_PAGESIZE;
  pBuffer += EEPROM_PAGESIZE;
}
if(numofsingle != 0) {
  /* Store the number of data to be written */
  dataindex = numofsingle;
  status = BSP_EEPROM_WritePage(pBuffer,
if (status != EEPROM_OK) {
    return status;
}
}

/* If all operations OK, return EEPROM_OK (0) */
return EEPROM_OK;

/**
* @brief Wait for EEPROM Standby state.
* @note This function allows to wait and check that EEPROM has finished the last operation. It is mostly used after Write operation: after receiving the buffer to be written, the EEPROM may need additional time to actually perform the write operation. During this time, it doesn't answer to I2C packets addressed to it. Once the write operation is complete, the EEPROM responds to its addresses.
* @retval EEPROM_OK (0) if operation is correctly performed, else return value different from EEPROM_OK (0) or the timeout user callback.
*/
uint32_t BSP_EEPROM_WaitEepromStandbyState(void)
if (EEPROM_IO_IsDeviceReady(EEPROMAddress, EEPROM_MAX_TRIALS) != HAL_OK)
{
    /* If the maximum number of trials has been reached, exit the function */
    BSP_EEPROM_TIMEOUT_UserCallback();
    return EEPROM_TIMEOUT;
}

return EEPROM_OK;

/**
 * @brief Basic management of the timeout situation.
 * @retval None
 */
__weak void BSP_EEPROM_TIMEOUT_UserCallback(void)
{
}
stm32f769i_discovery_sdram.h

Go to the documentation of this file.

```c
/**
 * @file       stm32f769i_discovery_sdram.h
 * @author     MCD Application Team
 * @version    V2.0.0
 * @date       30-December-2016
 * @brief      This file contains the common defines and functions prototypes for
 *             the stm32f769i_discovery_sdram.c driver.
 */
```

* @attention

```
<h2><center>&copy; COPYRIGHT(c) 2016 STM
microelectronics</center></h2>
```

* Redistribution and use in source and binary forms, with or without modification,
  are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
/* Define to prevent recursive inclusion ---------------------------------*/
#ifndef __STM32F769I_DISCOVERY_SDRAM_H
#define __STM32F769I_DISCOVERY_SDRAM_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes ---------------------------------*/
#include "stm32f7xx_hal.h"

/** @addtogroup BSP *
* @{
*/

/** @addtogroup STM32F769I_DISCOVERY *
* @{
*/

/** @addtogroup STM32F769I_DISCOVERY_SDRAM *
* @{
*/

/** @defgroup STM32F769I_DISCOVERY_SDRAM_Exp
* SDRAM Exported Types
*
* @{
*/

/** @brief SDRAM status structure definition
* @{
*/

/** @} */

/** @} */

/** @} */

/** @} */

#endif
#endif

```c
#define SDRAM_OK     ((uint8_t)0x00)
#define SDRAM_ERROR  ((uint8_t)0x01)

/**
 * @defgroup STM32F769I_DISCOVERY_SDRAM_Exported_Constants SDRAM Exported Constants
 *
 */

#define SDRAM_DEVICE_ADDR ((uint32_t)0xC0000000)
#define SDRAM_DEVICE_SIZE ((uint32_t)0x10000000) /* SDRAM device size in MBytes */

#define SDRAM_MEMORY_WIDTH FMC_SDRAM_MEM_BUS_WIDTH_32
#define SDCLOCK_PERIOD FMC_SDRAM_CLOCK_PERIOD_2
#define REFRESH_COUNT   ((uint32_t)0x0603) /* SDRAM refresh counter (100Mhz SD clock) */
#define SDRAM_TIMEOUT (uint32_t)0xFFFF /* DMA definitions for SDRAM DMA transfer */
#define __DMAx_CLK_ENABLE HAL_RCC_DMA2_CLK_ENABLE
```
#define __DMAx_CLK_DISABLE
HAL_RCC_DMA2_CLK_DISABLE

#define SDRAM_DMAx_CHANNEL DM_A_CHANNEL_0
#define SDRAM_DMAx_STREAM DM_A2_Stream0
#define SDRAM_DMAx_IRQn DM_A2_Stream0_IRQn
#define BSP_SDRAM_DMA_IRQHandler DM_A2_Stream0_IRQHandler

/**
 * @}
 */

/**
 * @brief FMC SDRAM Mode definition register defines
 */

#define SDRAM_MODEREG_BURST_LENGTH_1 ((uint16_t)0x0000)
#define SDRAM_MODEREG_BURST_LENGTH_2 ((uint16_t)0x0001)
#define SDRAM_MODEREG_BURST_LENGTH_4 ((uint16_t)0x0002)
#define SDRAM_MODEREG_BURST_LENGTH_8 ((uint16_t)0x0004)
#define SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL ((uint16_t)0x0000)
#define SDRAM_MODEREG_BURST_TYPE_INTERLEAVED ((uint16_t)0x0008)
#define SDRAM_MODEREG_CAS_LATENCY_2 ((uint16_t)0x0020)
#define SDRAM_MODEREG_CAS_LATENCY_3 ((uint16_t)0x0030)
#define SDRAM_MODEREG_OPERATING_MODE_STANDARD ((uint16_t)0x0000)
#define SDRAM_MODEREG_WRITEBURST_MODE_PROGRAMMED ((uint16_t)0x0000)
MMED ((uint16_t)0x0000)
00114 #define SDRAM_MODEREG_WRITEBURST_MODE_SINGLE ((uint16_t)0x0200)
00115 /**<
00116 * @}
00117 */
00118
00119 /**< @defgroup STM32F769I_DISCOVERY_SDRAM_Exported_Macro  SDRAM Exported Macro
00120 * @{
00121 */
00122 /**<
00123 * @}
00124 */
00125
00126 /**< @defgroup STM32F769I_DISCOVERY_SDRAM_Exported_Functions SDRAM Exported Functions
00127 * @{
00128 */
00129 uint8_t BSP_SDRAM_Init(void);
00130 uint8_t BSP_SDRAM_DeInit(void);
00131 void BSP_SDRAM_Initialization_sequence(uint32_t RefreshCount);
00132 uint8_t BSP_SDRAM_ReadData(uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize);
00133 uint8_t BSP_SDRAM_ReadData_DMA(uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize);
00134 uint8_t BSP_SDRAM_WriteData(uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize);
00135 uint8_t BSP_SDRAM_WriteData_DMA(uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize);
00136 uint8_t BSP_SDRAM_Sendcmd(FMC_SDRAM_CommandTypeDef *SdramCmd);
00137
00138 /* These functions can be modified in case the current settings (e.g. DMA stream)
need to be changed for specific application needs */

void BSP_SDRAM_MspInit(SDRAM_HandleTypeDef *hsdram, void *Params);

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

/**
 * @} */

/**
 * @} */

/**
 * @} */

/**
 * @} */

#else __cplusplus
}
#endif /*__STM32F769I_DISCOVERY_SDRAM_H*/

/**************************** (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
stm32f769i_discovery_sd.h

Go to the documentation of this file.

```c
/**
 * @file stm32f769i_discovery_sd.h
 * @author MCD Application Team
 * @version V2.0.0
 * @date 30-December-2016
 * @brief This file contains the common defines and functions prototypes for
 *        the stm32f769i_discovery_sd.c driver.
 */

* @attention

* <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>

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

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
 CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
/*
  Define to prevent recursive inclusion ---------------------------*/
#ifndef __STM32F769I_DISCOVERY_SD_H
#define __STM32F769I_DISCOVERY_SD_H
#ifdef __cplusplus
extern "C" {
#endif

/* Includes -----------------------------------------------*/
#include "stm32f769i_discovery.h"

/** @addtogroup BSP *
 * @{
 */

/** @addtogroup STM32F769I_DISCOVERY *
 * @{
 */

/** @addtogroup STM32F769I_DISCOVERY_SD *
 * @{
 */

/** @defgroup STM32F769I_DISCOVERY_SD_Export *
 * @{
 */

/** @} */

/** @} */

/** @} */

/** @} */
* @brief SD Card information structure
*/
#define BSP_SD_CardInfo HAL_SD_CardInfoTypeDef
/**
 * @}
 */

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

/**
 * @brief SD transfer state definition
*/
#define SD_TRANSFER_OK ((uint8_t)0x00)
#define SD_TRANSFER_BUSY ((uint8_t)0x01)

/** @defgroup STM32F769I_DISCOVERY_SD_Exported_Constants SD Exported Constants */
#define SD_PRESENT ((uint8_t)0x01)
#define SD_NOT_PRESENT ((uint8_t)0x00)
#define SD_DATATIMEOUT ((uint32_t)100000000)
/* DMA definitions for SD DMA transfer */

#define __DMAx_TxRx_CLK_ENABLE
    HAL_RCC_DMA2_CLK_ENABLE

#define SD_DMAx_Tx_CHANNEL
    DMA_CHANNEL_11
#define SD_DMAx_Rx_CHANNEL
    DMA_CHANNEL_11
#define SD_DMAx_Tx_STREAM
    DMA2_Stream5
#define SD_DMAx_Rx_STREAM
    DMA2_Stream0
#define SD_DMAx_Tx_IRQn
    DMA2_Stream5_IRQn
#define SD_DMAx_Rx_IRQn
    DMA2_Stream0_IRQn
#define BSP_SDMMC_IRQHandler
    MMC2_IRQHandler
#define BSP_SDMMC_DMA_Tx_IRQHandler
    DMA2_Stream5_IRQHandler
#define BSP_SDMMC_DMA_Rx_IRQHandler
    DMA2_Stream0_IRQHandler
#define SD_DetectIRQHandler()
    HAL_GPIO_EXTI_IRQHandler(SD_DETECT_PIN)

/**
 * @}
 */

/**
 * @defgroup STM32F769I_DISCOVERY_SD_Export
 STM32F769I Discovery SD Exported Macro
 */

/**
 * @defgroup STM32F769I_DISCOVERY_SD_Export
 */

/* @} */

/**
 * @defgroup STM32F769I_DISCOVERY_SD_Export
 */

/* @} */
/**	@defgroup STM32F769I_DISCOVERY_SD_Exported_Functions STM32F769I Discovery SD Exported Functions
*/

uint8_t BSP_SD_Init(void);
uint8_t BSP_SD_DeInit(void);
uint8_t BSP_SD_ITConfig(void);
uint8_t BSP_SD_ReadBlocks(uint32_t *pData, uint32_t ReadAddr, uint32_t NumOfBlocks, uint32_t Timeout);
uint8_t BSP_SD_WriteBlocks(uint32_t *pData, uint32_t WriteAddr, uint32_t NumOfBlocks, uint32_t Timeout);
uint8_t BSP_SD_ReadBlocks_DMA(uint32_t *pData, uint32_t ReadAddr, uint32_t NumOfBlocks);
uint8_t BSP_SD_WriteBlocks_DMA(uint32_t *pData, uint32_t WriteAddr, uint32_t NumOfBlocks);
uint8_t BSP_SD_Erase(uint32_t StartAddr, uint32_t EndAddr);
uint8_t BSP_SD_GetCardState(void);
void BSP_SD_GetCardInfo(HAL_SD_CardInfoTypeDef *CardInfo);
uint8_t BSP_SD_IsDetected(void);

/* These functions can be modified in case the current settings (e.g. DMA stream)
need to be changed for specific application needs */
void BSP_SD_MspInit(SD_HandleTypeDef *hsd, void *Params);
void BSP_SD_Detect_MspInit(SD_HandleTypeDef *hsd, void *Params);
void BSP_SD_MspDeInit(SD_HandleTypeDef *hsd, void *Params);
void BSP_SD_AbortCallback(void);
void BSP_SD_WriteCpltCallback(void);

void BSP_SD_ReadCpltCallback(void);

/****
	*
	*/

/****
	*
	*/

/****
	*
	*/

#endif

#ifdef __cplusplus
}
#endif

/*	__STM32F769I_DISCOVERY_SD_H	*/

/****************************	(C)	COPYRIGHT	STMicroelectronics	*****END	OF	FILE****/
stm32f769i_discovery_sd.c

Go to the documentation of this file.

```c
/**
 * @file  stm32f769i_discovery_sd.c
 * @author MCD Application Team
 * @version V2.0.0
 * @date  30-December-2016
 * @brief This file includes the uSD card driver mounted on STM32F769I-Discovery board.
 * 
 * @attention
 * <h2><center>&copy; COPYRIGHT(c) 2016 STM microelectronics</center></h2>
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
 * 1. Redistributions of source code must
retain the above copyright notice,
* this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright notice,
* this list of conditions and the following disclaimer in the documentation
* and/or other materials provided with the distribution.
* 3. Neither the name of STMicroelectronics nor the names of its contributors
* may be used to endorse or promote products derived from this software
* without specific prior written permission.
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
* DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
* SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
* CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
* OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
1. How To use this driver:

- This driver is used to drive the micro SD external card mounted on STM32F769I-Discovery board.
- This driver does not need a specific component driver for the micro SD device to be included with.

2. Driver description:

Initialization steps:

- Initialize the micro SD card using the BSP_SD_Init() function. This function includes the MSP layer hardware resources initialization and the SDIO interface configuration to interface with the external micro SD. It also includes the micro SD initialization sequence.
- To check the SD card presence you can use the function BSP_SD_IsDetected() which returns the detection status.
- If SD presence detection interrupt mode is desired, you must configure the SD detection interrupt mode by calling the function BSP_SD_ITConfig(). The interrupt is generated as an external interrupt whenever the micro SD card is...
plugged/unplugged in/from the discovery board.

- The function BSP_SD_GetCardInfo() is used to get the micro SD card information which is stored in the structure "HAL_SD_CardInfoTypedef".

+ Micro SD card operations

- The micro SD card can be accessed with read/write block(s) operations once it is ready for access. The access can be performed whether using the polling mode by calling the functions BSP_SD_ReadBlocks()/BSP_SD_WriteBlocks(), or by DMA transfer using the functions BSP_SD_ReadBlocks_DMA()/BSP_SD_WriteBlocks_DMA()

- The DMA transfer complete is used with interrupt mode. Once the SD transfer is complete, the SD interrupt is handled using the function BSP_SD_IRQHandler(), the DMA Tx/Rx transfer complete are handled using the functions BSP_SD_DMA_Tx_IRQHandler()/BSP_SD_DMA_Rx_IRQHandler(). The corresponding user callbacks are implemented by the user at application level.

- 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_GetCardState().
#include "stm32f769i_discovery_sd.h"

/**
 * @addtogroup BSP
 * @{
 */

/**
 * @addtogroup STM32F769I_DISCOVERY
 * @{
 */

/**
 * @defgroup STM32F769I_DISCOVERY_SD STM32F769I_DISCOVERY_SD
 * @{
 */

/**
 * @defgroup STM32F769I_DISCOVERY_SD_Private_TypesDefinitions STM32F769I Discovery Sd Private TypesDefinitions
 * @{
 */

/**
 * @defgroup STM32F769I_DISCOVERY_SD_Private_Definedes STM32F769I Discovery Sd Private Defines
 * @{
 */

/**
 * @defgroup STM32F769I_DISCOVERY_SD_Private_Macros STM32F769I Discovery Sd Private Macro
 * @{
 */

/**
 * @addtogroup BSP
 */

/**
 * @addtogroup STM32F769I_DISCOVERY
 */

/**
 * @addtogroup STM32F769I_DISCOVERY_SD
 */

/**
 * @addtogroup STM32F769I_DISCOVERY_SD_Private_TypesDefinitions
 */

/**
 * @addtogroup STM32F769I_DISCOVERY_SD_Private_Definedes
 */

/**
 * @addtogroup STM32F769I_DISCOVERY_SD_Private_Macros
 */
/**
* @defgroup STM32F769I_DISCOVERY_SD_Private_Variables STM32F769I Discovery Sd Private Variables
* @{
* @}
* @}
*/

SD_HandleTypeDef uSdHandle;

/**
* @defgroup STM32F769I_DISCOVERY_SD_Private_FunctionPrototypes STM32F769I Discovery Sd Private Prototypes
* @{
* @}
* @}
*/

/**
* @defgroup STM32F769I_DISCOVERY_SD_Private_Functions STM32F769I Discovery Sd Private Functions
* @{
* @}
* @}
*/

/**
* @brief Initializes the SD card device.
* @retval SD status
*/
uint8_t BSP_SD_Init(void)
{
    uint8_t sd_state = MSD_OK;
}
/* PLLSAI is dedicated to LCD periph. Do not use it to get 48MHz*/

/* uSD device interface configuration */
#define uSdHandle.Instance SDMMC2;
#define uSdHandle.Init.ClockEdge SDMC_CLOCK_EDGE_RISING;
#define uSdHandle.Init.ClockBypass SDMC_CLOCK_BYPASS_DISABLE;
#define uSdHandle.Init.ClockPowerSave SDMC_CLOCK_POWER_SAVE_DISABLE;
#define uSdHandle.Init.BusWide SDMC_BUS_WIDE_1B;
#define uSdHandle.Init.HardwareFlowControl SDMC_HARDWARE_FLOW_CONTROL_DISABLE;
#define uSdHandle.Init.ClockDiv SDMC_TRANSFER_CLK_DIV;

/* Msp SD Detect pin initialization */
BSP_SD_Detect_MspInit(&uSdHandle, NULL);
if (BSP_SD_IsDetected() != SD_PRESENT) /* Check if SD card is present */
{
    return MSD_ERROR_SD_NOT_PRESENT;
}

/* Msp SD initialization */
BSP_SD_MspInit(&uSdHandle, NULL);

/* HAL SD initialization */
if (HAL_SD_Init(&uSdHandle) != HAL_OK)
{
    sd_state = MSD_ERROR;
}

/* Configure SD Bus width */
if (sd_state == MSD_OK)
{
    /* Enable wide operation */
    if (HAL_SD_ConfigWideBusOperation(&uSdHandle, SDMMC_BUS_WIDE_4B) != HAL_OK)
    {
        sd_state = MSD_ERROR;
    } else
    {
        sd_state = MSD_OK;
    }
}

return sd_state;

/**
 * @brief DeInitializes the SD card device.
 *
 * @retval SD status
 */
uint8_t BSP_SD_DeInit(void)
{
    uint8_t sd_state = MSD_OK;
    uSdHandle.Instance = SDMMC2;
    /* HAL SD deinitialization */
    if (HAL_SD_DeInit(&uSdHandle) != HAL_OK)
    {
        sd_state = MSD_ERROR;
    }

    /* Msp SD deinitialization */
    uSdHandle.Instance = SDMMC2;
    BSP_SD_MspDeInit(&uSdHandle, NULL);
return sd_state;
}

/**
 * @brief Configures Interrupt mode for SD detection pin.
 * @retval Returns 0
 */
uint8_t BSP_SD_ITConfig(void)
{
    GPIO_InitTypeDef gpio_init_structure;

    /* Configure Interrupt mode for SD detection pin */
    gpio_init_structure.Pin = SD_DETECT_PIN;
    gpio_init_structure.Pull = GPIO_PULLUP;
    gpio_init_structure.Speed = GPIO_SPEED_FAST;
    gpio_init_structure.Mode = GPIO_MODE_IT_RISING_FALLING;
    HAL_GPIO_Init(SD_DETECT_GPIO_PORT, &gpio_init_structure);

    /* Enable and set SD detect EXTI Interrupt to the lowest priority */
    HAL_NVIC_SetPriority((IRQn_Type)(SD_DETECT_EXTI_IRQHandler), 0x0F, 0x00);
    HAL_NVIC_EnableIRQ((IRQn_Type)(SD_DETECT_EXTI_IRQHandler));

    return MSD_OK;
}
uint8_t BSP_SD_IsDetected(void) {
	__IO uint8_t status = SD_PRESENT;

/* Check SD card detect pin */
if (HAL_GPIO_ReadPin(SD_DETECT_GPIO_PORT, SD_DETECT_PIN) == GPIO_PIN_SET) {
	status = SD_NOT_PRESENT;
}
return status;
}

/**
 * @brief Reads block(s) from a specified address in an SD card, in polling mode.
 * @param pData: Pointer to the buffer that will contain the data to transmit
 * @param ReadAddr: Address from where data is to be read
 * @param NumOfBlocks: Number of SD blocks to read
 * @param Timeout: Timeout for read operation
 * @retval SD status
 */
uint8_t BSP_SD_ReadBlocks(uint32_t *pData, uint32_t ReadAddr, uint32_t NumOfBlocks, uint32_t Timeout) {
	if(HAL_SD_ReadBlocks(&uSdHandle, (uint8_t*)pData, ReadAddr, NumOfBlocks, Timeout) != HAL_OK ) {
		return MSD_ERROR;
}
else
{
    return MSD_OK;
}
}
#endif

/**
* @brief Writes block(s) to a specified address in an SD card, in polling mode.
* @param pData: Pointer to the buffer that will contain the data to transmit
* @param WriteAddr: Address from where data is to be written
* @param NumOfBlocks: Number of SD blocks to write
* @param Timeout: Timeout for write operation
* @retval SD status
*/
uint8_t BSP_SD_WriteBlocks(uint32_t *pData, uint32_t WriteAddr, uint32_t NumOfBlocks, uint32_t Timeout)
{
    if(HAL_SD_WriteBlocks(&uSdHandle, (uint8_t *)pData, WriteAddr, NumOfBlocks, Timeout) != HAL_OK)
    {
        return MSD_ERROR;
    }
    else
    {
        return MSD_OK;
    }
}
@brief Reads block(s) from a specified address in an SD card, in DMA mode.
@param pData: Pointer to the buffer that will contain the data to transmit
@param ReadAddr: Address from where data is to be read
@param NumOfBlocks: Number of SD blocks to read
@retval SD status

uint8_t BSP_SD_ReadBlocks_DMA(uint32_t *pData, uint32_t ReadAddr, uint32_t NumOfBlocks)
{
    /* Read block(s) in DMA transfer mode */
    if(HAL_SD_ReadBlocks_DMA(&uSdHandle, (uint8_t *)pData, ReadAddr, NumOfBlocks) != HAL_OK)
    {
        return MSD_ERROR;
    }
    else
    {
        return MSD_OK;
    }
}

@brief Writes block(s) to a specified address in an SD card, in DMA mode.
@param pData: Pointer to the buffer that will contain the data to transmit
@param WriteAddr: Address from where data is to be written
@param NumOfBlocks: Number of SD blocks to write
@retval SD status

uint8_t BSP_SD_WriteBlocks_DMA(uint32_t *pData, uint32_t WriteAddr, uint32_t NumOfBlocks)


```c
/
/* Write block(s) in DMA transfer mode */
if(HAL_SD_WriteBlocks_DMA(&uSdHandle, (uint8_t*)pData, WriteAddr, NumOfBlocks) != HAL_OK)
{
    return MSD_ERROR;
}
else
{
    return MSD_OK;
}

/**
 * @brief Erases the specified memory area of the given SD card.
 * @param StartAddr: Start byte address
 * @param EndAddr: End byte address
 * @retval SD status
 */
uint8_t BSP_SD_Erase(uint32_t StartAddr, uint32_t EndAddr)
{
    if(HAL_SD_Erase(&uSdHandle, StartAddr, EndAddr) != HAL_OK)
    {
        return MSD_ERROR;
    }
    else
    {
        return MSD_OK;
    }

    /*@brief Initializes the SD MSP.*/
```
__weak void BSP_SD_MspInit(SD_HandleTypeDef *hsd, void *Params) {
  static DMA_HandleTypeDef dma_rx_handle;
  static DMA_HandleTypeDef dma_tx_handle;
  GPIO_InitTypeDef gpio_init_structure;

  /* Enable SDMMC2 clock */
  __HAL_RCC_SDMMC2_CLK_ENABLE();

  /* Enable DMA2 clocks */
  __DMAx_TxRx_CLK_ENABLE();

  /* Enable GPIOs clock */
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOG_CLK_ENABLE();

  /* Common GPIO configuration */
  gpio_init_structure.Mode = GPIO_MODE_AF_PP;
  gpio_init_structure.Pull = GPIO_PULLUP;
  gpio_init_structure.Speed = GPIO_SPEED_HIGH;

  /* GPIOB configuration */
  gpio_init_structure.Alternate = GPIO_AF10_SDMMC2;
  gpio_init_structure.Pin = GPIO_PIN_3 | GPIO_PIN_4;
  HAL_GPIO_Init(GPIOB, &gpio_init_structure);
/* GPIOD configuration */
gpio_init_structure.Alternate = GPIO_AF11_SDMMC2;
gpio_init_structure.Pin = GPIO_PIN_6 | GPIO_PIN_7;
HAL_GPIO_Init(GPIOD, &gpio_init_structure);

/* GPIOG configuration */
gpio_init_structure.Pin = GPIO_PIN_9 | GPIO_PIN_10;
HAL_GPIO_Init(GPIOG, &gpio_init_structure);

/* NVIC configuration for SDMMC2 interrupts */
HAL_NVIC_SetPriority(SDMMC2_IRQn, 0x0E, 0);
HAL_NVIC_EnableIRQ(SDMMC2_IRQn);

/* Configure DMA Rx parameters */
dma_rx_handle.Init.Channel = SD_DMAx_Rx_CHANNEL;
dma_rx_handle.Init.Direction = DMA_PERIPH_TO_MEMORY;
dma_rx_handle.Init.PeriphInc = DMA_PINC_DISABLE;
dma_rx_handle.Init.MemInc = DMA_MINC_ENABLE;
dma_rx_handle.Init.PeriphDataAlignment = DMA_PDATAALIGN_WORD;
dma_rx_handle.Init.MemDataAlignment = DMA_MDATAALIGN_WORD;
dma_rx_handle.Init.Mode = DMA_PFCTRL;
dma_rx_handle.Init.Priority = DMA_PRIORITY_MOST_
MA_PRIORITY_VERY_HIGH;
00402 dma_rx_handle.Init.FIFOMode = D
MA_FIFOMODE_ENABLE;
00403 dma_rx_handle.Init.FIFOThreshold = D
MA_FIFO_THRESHOLD_FULL;
00404 dma_rx_handle.Init.MemBurst = D
MA_MBURST_INC4;
00405 dma_rx_handle.Init.PeriphBurst = D
MA_PBURST_INC4;
00406
00407 dma_rx_handle.Instance = SD_DMAx_Rx_STREAM;
00408
00409 /* Associate the DMA handle */
00410 __HAL_LINKDMA(hsd, hdmarsx, dma_rx_handle);
00411
00412 /* Deinitialize the stream for new transfer */
00413 HAL_DMA_DeInit(&dma_rx_handle);
00414
00415 /* Configure the DMA stream */
00416 HAL_DMA_Init(&dma_rx_handle);
00417
00418 /* Configure DMA Tx parameters */
00419 dma_tx_handle.Init.Channel = SD_DMAx_Tx_CHANNEL;
00420 dma_tx_handle.Init.Direction = D
MA_MEMORY_TO_PERIPH;
00421 dma_tx_handle.Init.PeriphInc = D
MA_PINC_DISABLE;
00422 dma_tx_handle.Init.MemInc = D
MA_MINC_ENABLE;
00423 dma_tx_handle.Init.PeriphDataAlignment = D
MA_PDATALIGN_WORD;
00424 dma_tx_handle.Init.MemDataAlignment = D
MA_MDATAALIGN_WORD;
00425 dma_tx_handle.Init.Mode = D
MA_PFCTRL;
00426  dma_tx_handle.Init.Priority = D
MA_PRIORITY_VERY_HIGH;
00427  dma_tx_handle.Init.FIFOMode = D
MA_FIFOMODE_ENABLE;
00428  dma_tx_handle.Init.FIFOThreshold = D
MA_FIFO_THRESHOLD_FULL;
00429  dma_tx_handle.Init.MemBurst = D
MA_MBURST_INC4;
00430  dma_tx_handle.Init.PeriphBurst = D
MA_PBURST_INC4;
00431
00432  dma_tx_handle.Instance = SD_DMAx_Tx_STREAM;
00433
00434  /* Associate the DMA handle */
00435  __HAL_LINKDMA(hsd, hdmatx, dma_tx_handle);
00436
00437  /* Deinitialize the stream for new transfer */
00438  HAL_DMA_DeInit(&dma_tx_handle);
00439
00440  /* Configure the DMA stream */
00441  HAL_DMA_Init(&dma_tx_handle);
00442
00443  /* NVIC configuration for DMA transfer complete interrupt */
00444  HAL_NVIC_SetPriority(SD_DMAx_Rx_IRQn, 0x0F, 0);
00445  HAL_NVIC_EnableIRQ(SD_DMAx_Rx_IRQn);
00446
00447  /* NVIC configuration for DMA transfer complete interrupt */
00448  HAL_NVIC_SetPriority(SD_DMAx_Tx_IRQn, 0x0F, 0);
00449  HAL_NVIC_EnableIRQ(SD_DMAx_Tx_IRQn);
00450 }
void BSP_SD_Detect_MspInit(SD_HandleTypeDef *hsd, void *Params)
{
    GPIO_InitTypeDef gpio_init_structure;

    SD_DETECT_GPIO_CLK_ENABLE();

    /* GPIO configuration in input for uSD_Detect signal */
    gpio_init_structure.Pin = SD_DETECT_PIN;
    gpio_init_structure.Mode = GPIO_MODE_INPUT;
    gpio_init_structure.Pull = GPIO_PULLUP;
    gpio_init_structure.Speed = GPIO_SPEED_HIGH;
    HAL_GPIO_Init(SD_DETECT_GPIO_PORT, &gpio_init_structure);
}

/* @brief DeInitializes the SD MSP. */
void BSP_SD_MspDeInit(SD_HandleTypeDef *hsd, void *Params)
f *hsd, void *Params) {
    static DMA_HandleTypeDef dma_rx_handle;
    static DMA_HandleTypeDef dma_tx_handle;

    /* Disable NVIC for DMA transfer complete interrupts */
    HAL_NVIC_DisableIRQ(SD_DMAx_Rx_IRQn);
    HAL_NVIC_DisableIRQ(SD_DMAx_Tx_IRQn);

    /* Deinitialize the stream for new transfer */
    dma_rx_handle.Instance = SD_DMAx_Rx_STREAM;
    HAL_DMA_DeInit(&dma_rx_handle);

    /* Deinitialize the stream for new transfer */
    dma_tx_handle.Instance = SD_DMAx_Tx_STREAM;
    HAL_DMA_DeInit(&dma_tx_handle);

    /* Disable NVIC for SDIO interrupts */
    HAL_NVIC_DisableIRQ(SDIO_IRQn);

    /* DeInit GPIO pins can be done in the application (by surcharging this __weak function) */

    /* Disable SDIO clock */
    __HAL_RCC_SDIO_CLK_DISABLE();

    /* GPOI pins clock and DMA cloks can be shut down in the application */
    by surcgarging this __weak function */
```c
/**
 * @brief Gets the current SD card data status.
 * @retval Data transfer state.
 * This value can be one of the following values:
 * @arg SD_TRANSFER_OK: No data transfer is acting
 * @arg SD_TRANSFER_BUSY: Data transfer is acting
 */

uint8_t BSP_SD_GetCardState(void)
{
    return ((HAL_SD_GetCardState(&uSdHandle) == HAL_SD_CARD_TRANSFER) ? SD_TRANSFER_OK : SD_TRANSFER_BUSY);
}

/**
 * @brief Get SD information about specific SD card.
 * @param CardInfo: Pointer to HAL_SD_CardInfoTypeDef structure
 * @retval None
 */

void BSP_SD_GetCardInfo(HAL_SD_CardInfoTypeDef *CardInfo)
{
    /* Get SD card Information */
    HAL_SD_GetCardInfo(&uSdHandle, CardInfo);
}

/**
 * @brief SD Abort callbacks
 */
```
void HAL_SD_AbortCallback(SD_HandleTypeDef *hsd) {
    BSP_SD_AbortCallback();
}

/**
* @brief Tx Transfer completed callbacks
* @param hsd: SD handle
* @retval None
*/

void HAL_SD_TxCpltCallback(SD_HandleTypeDef *hsd) {
    BSP_SD_WriteCpltCallback();
}

/**
* @brief Rx Transfer completed callbacks
* @param hsd: SD handle
* @retval None
*/

void HAL_SD_RxCpltCallback(SD_HandleTypeDef *hsd) {
    BSP_SD_ReadCpltCallback();
}

/**
* @brief BSP SD Abort callbacks
* @retval None
*/

__weak void BSP_SD_AbortCallback(void) {

/**
 * @brief BSP Tx Transfer completed callbacks
 * @retval None
 */
__weak void BSP_SD_WriteCpltCallback(void)
{
}

/**
 * @brief BSP Rx Transfer completed callbacks
 * @retval None
 */
__weak void BSP_SD_ReadCpltCallback(void)
{
}
stm32f769i_discovery_sdram.c

Go to the documentation of this file.

```c
/**
 * @file     stm32f769i_discovery_sdram.c
 * @author   MCD Application Team
 * @version  V2.0.0
 * @date     30-December-2016
 * @brief    This file includes the SDRAM driver for the MT48LC4M32B2B5-6A memory device mounted on STM32F769I-DISCOVERY boards.
 * @verbatim
 * How To use this driver:
 * ------------------------
 * - This driver is used to drive the MT48LC4M32B2B5-6A SDRAM external memory mounted on STM32F769I-DISCOVERY board.
 * - This driver does not need a specific component driver for the SDRAM device to be included with.
 * @verbatim
 * Driver description:
```
+ Initialization steps:
  o Initialize the SDRAM external memory using the BSP_SDRAM_Init() function. This function includes the MSP layer hardware resources initialization and the FMC controller configuration to interface with the external SDRAM memory.
  o It contains the SDRAM initialization sequence to program the SDRAM external device using the function BSP_SDRAM_Initialization_sequence(). Note that this sequence is standard for all SDRAM devices, but can include some differences from a device to another. If it is the case, the right sequence should be implemented separately.

+ SDRAM read/write operations
  o SDRAM external memory can be accessed with read/write operations once it is initialized.
  o Read/write operation can be performed with AHB access using the functions BSP_SDRAM_ReadData()/BSP_SDRAM_WriteData(), or by DMA transfer using the functions BSP_SDRAM_ReadData_DMA()/BSP_SDRAM_WriteData_DMA().
  o The AHB access is performed with 32-bit width transaction, the DMA transfer configuration is fixed at single (no burst) word transfer (see the SDRAM_MspInit() static function).
  o User can implement his own functions for read/write access with his desired configurations.
  o If interrupt mode is used for DMA tra
nsfer, the function BSP_SDRAM_DMA_IRQHandler() is called in IRQ handler file, to serve the generated interrupt once the DMA transfer is complete. You can send a command to the SDRAM device in runtime using the function BSP_SDRAM_Sendcmd(), and giving the desired command as parameter chosen between the predefined commands of the "FMC_SDRAM_CommandTypeDef" structure.

@endverbatim

******************************************
************************************
* @attention
*
* <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>
*
* Redistribution and use in source and binary forms, with or without modification,
* are permitted provided that the following conditions are met:
* 1. Redistributions of source code must retain the above copyright notice,
* this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright notice,
* this list of conditions and the following disclaimer in the documentation
* and/or other materials provided with the distribution.
* 3. Neither the name of STMicroelectronics nor the names of its contributors
* may be used to endorse or promote products derived from this software
* without specific prior written perm
ission.
00062  *
00063  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
00064  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
00065  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00066  * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
00067  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
00068  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
00069  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
00070  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
00071  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00072  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00073  *
00074 ***********************************************************************************************
00075  */
00076
00077 /* Includes ..........................................................*/
00078 #include "stm32f769i_discovery_sdram.h"
00079
00080 /** @addtogroup BSP
00081  *
00082  */
00083
00084 /** @addtogroup STM32F769I_DISCOVERY
00085  */
/**
 * \defgroup STM32F769I_DISCOVERY_SDRAM STM32F769I_DISCOVERY SDRAM
 */

/**
 * \defgroup STM32F769I_DISCOVERY_SDRAM_Private_Types_Definitions SDRAM Private Types Definitions
 */

/**
 * \defgroup STM32F769I_DISCOVERY_SDRAM_Private_Defines SDRAM Private Defines
 */

/**
 * \defgroup STM32F769I_DISCOVERY_SDRAM_Private_Macros SDRAM Private Macros
 */

/**
 * \defgroup STM32F769I_DISCOVERY_SDRAM_Private_Variables SDRAM Private Variables
 */

SDRAM_HandleTypeDef sdramHandle;
static FMC_SDRAM_TimingTypeDef Timing;

static FMC_SDRAM_CommandTypeDef Command;

/**
 * @}
 */

/**
 * @defgroup STM32F769I_DISCOVERY_SDRAM_Private_Function_Prototypes SDRAM Private Function Prototypes
 * @{
 */

/**
 * @defgroup STM32F769I_DISCOVERY_SDRAM_Private_Functions SDRAM Private Functions
 * @{
 */

/**
 * @brief Initializes the SDRAM device.
 * @retval SDRAM status
 */
uint8_t BSP_SDRAM_Init(void)
{
  static uint8_t sdramstatus = SDRAM_ERROR;
  /* SDRAM device configuration */
  sdramHandle.Instance = FMC_SDRAM_DEVICE;
  /* Timing configuration for 100Mhz as SDRAM clock frequency (System clock is up to 200Mhz) */
  Timing.LoadToActiveDelay = 2;
  Timing.ExitSelfRefreshDelay = 7;
  Timing.SelfRefreshTime = 4;
  Timing.RowCycleDelay = 7;
Timing.WriteRecoveryTime = 2;
Timing.RPDelay = 2;
Timing.RCDDelay = 2;
sdramHandle.Init.SDBank = FMC_SDRAM_BANK1;
sdramHandle.Init.ColumnBitsNumber = FMC_SDRAM_COLUMN_BITS_NUM_8;
sdramHandle.Init.RowBitsNumber = FMC_SDRAM_ROW_BITS_NUM_12;
sdramHandle.Init.MemoryDataWidth = SDRAM_MEMORY_WIDTH;
sdramHandle.Init.InternalBankNumber = FMC_SDRAM_INTERN_BANKS_NUM_4;
sdramHandle.Init.CASLatency = FMC_SDRAM_CAS_LATENCY_3;
sdramHandle.Init.WriteProtection = FMC_SDRAM_WRITE_PROTECTION_DISABLE;
sdramHandle.Init.SDClockPeriod = SDCLK_PERIOD;
sdramHandle.Init.ReadBurst = FMC_SDRAM_RBURST_ENABLE;
sdramHandle.Init.ReadPipeDelay = FMC_SDRAM_RPIPE_DELAY_0;

/* SDRAM controller initialization */
BSP_SDRAM_MspInit(&sdramHandle, NULL); /* __weak function can be rewritten by the application */
if(HAL_SDRAM_Init(&sdramHandle, &Timing) != HAL_OK)
{
    sdramstatus = SDRAM_ERROR;
}
else


```c
00173   {
00174       sdramstatus = SDRAM_OK;
00175   }
00176
00177   /* SDRAM initialization sequence */
00178   BSP_SDRAM_Initialization_sequence(REFRESH_COUNT);
00179
00180   return sdramstatus;
00181 }
00182
00183 /**
00184   * @brief DeInitializes the SDRAM device.
00185   * @retval SDRAM status
00186   */
00187 uint8_t BSP_SDRAM_DeInit(void)
00188 {
00189     static uint8_t sdramstatus = SDRAM_ERROR;
00190     /* SDRAM device de-initialization */
00191     sdramHandle.Instance = FMC_SDRAM_DEVICE;
00192
00193     if(HAL_SDRAM_DeInit(&sdramHandle) != HAL_OK)
00194     {
00195         sdramstatus = SDRAM_ERROR;
00196     }
00197     else
00198     {
00199         sdramstatus = SDRAM_OK;
00200     }
00201
00202     /* SDRAM controller de-initialization */
00203     BSP_SDRAM_MspDeInit(&sdramHandle, NULL);
00204
00205     return sdramstatus;
00206 }
00207```
/**
 * @brief Programs the SDRAM device.
 * @param RefreshCount: SDRAM refresh counter value
 * @retval None
 */

void BSP_SDRAM_Initialization_sequence(uint32_t RefreshCount)
{
    __IO uint32_t tmpmrd = 0;
    /* Step 1: Configure a clock configuration enable command */
    Command.CommandMode = FMC_SDRAM_CMD_CLK_ENABLE;
    Command.CommandTarget = FMC_SDRAM_CMD_TARGET_BANK1;
    CommandАвтоRefreshNumber = 1;
    Command.ModeRegisterDefinition = 0;
    /* Send the command */
    HAL_SDRAM_SendCommand(&sdramHandle, &Command, SDRAM_TIMEOUT);

    /* Step 2: Insert 100 us minimum delay */
    /* Inserted delay is equal to 1 ms due to systick time base unit (ms) */
    HAL_Delay(1);

    /* Step 3: Configure a PALL (precharge all) command */
    Command.CommandMode = FMC_SDRAM_CMD_PALL;
    Command.CommandTarget = FMC_SDRAM_CMD_TARGET_BANK1;
    CommandАвтоRefreshNumber = 1;
    Command.ModeRegisterDefinition = 0;
/* Send the command */
HAL_SDRAM_SendCommand(&sdramHandle, &Command, SDRAM_TIMEOUT);

/* Step 4: Configure an Auto Refresh command */
Command.CommandMode = FMC_SDRAM_CMD_AUTOREFRESH_MODE;
Command.CommandTarget = FMC_SDRAM_CMD_TARGET_BANK1;
Command.AutoRefreshNumber = 8;
Command.ModeRegisterDefinition = 0;

/* Send the command */
HAL_SDRAM_SendCommand(&sdramHandle, &Command, SDRAM_TIMEOUT);

/* Step 5: Program the external memory mode register */
tmpmrd = (uint32_t)SDRAM_MODEREG_BURST_LEN_GTH_1 | SDRAM_MODEREG_BURST_TYP_E_SEQUENTIAL | SDRAM_MODEREG_CAS_LATEN_CY_3 | SDRAM_MODEREG_OPERATING_MODE_STANDARD | SDRAM_MODEREG_WRITEBURST_MODE_SINGLE;

Command.CommandMode = FMC_SDRAM_CMD_LOAD_MODE;
Command.CommandTarget = FMC_SDRAM_CMD_TARGET_BANK1;
Command.AutoRefreshNumber = 1;
Command.ModeRegisterDefinition = tmpmrd;
/* Send the command */
HAL_SDRAM_SendCommand(&sdramHandle, &Command, SDRAM_TIMEOUT);

/* Step 6: Set the refresh rate counter */
/* Set the device refresh rate */
HAL_SDRAM_ProgramRefreshRate(&sdramHandle, RefreshCount);
}

/** *
@brief Reads an amount of data from the SDRAM memory in polling mode.
@param uwStartAddress: Read start address
@param pData: Pointer to data to be read
@param uwDataSize: Size of read data from the memory
@retval SDRAM status */

uint8_t BSP_SDRAM_ReadData(uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)
{
    if(HAL_SDRAM_Read_32b(&sdramHandle, (uint32_t *)uwStartAddress, pData, uwDataSize) != HAL_OK )
    {
        return SDRAM_ERROR;
    }

    else
    {
        return SDRAM_OK;
    }
}
/**
 * @brief Reads an amount of data from the SDRAM memory in DMA mode.
 * @param uwStartAddress: Read start address
 * @param pData: Pointer to data to be read
 * @param uwDataSize: Size of read data from the memory
 * @retval SDRAM status
 */

uint8_t BSP_SDRAM_ReadData_DMA(uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize) {
    if(HAL_SDRAM_Read_DMA(&sdramHandle, (uint32_t *)uwStartAddress, pData, uwDataSize) != HAL_OK )
    {
        return SDRAM_ERROR;
    }
    else
    { return SDRAM_OK; }
}

/**
 * @brief Writes an amount of data to the SDRAM memory in polling mode.
 * @param uwStartAddress: Write start address
 * @param pData: Pointer to data to be written
 * @param uwDataSize: Size of written data from the memory
 * @retval SDRAM status
 */
```c
00313 uint8_t BSP_SDRAM_WriteData(uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)
00314 {
00315   if(HAL_SDRAM_Write_32b(&sdramHandle, (uint32_t *)uwStartAddress, pData, uwDataSize) != HAL_OK)
00316       { return SDRAM_ERROR; }
00317   } else
00318   { return SDRAM_OK; }
00319 }
00320
00321 /* @brief Writes an amount of data to the SDRAM memory in DMA mode.
00322   * @param uwStartAddress: Write start address
00323   * @param pData: Pointer to data to be written
00324   * @param uwDataSize: Size of written data from the memory
00325   * @retval SDRAM status */
00326 uint8_t BSP_SDRAM_WriteData_DMA(uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize)
00327 {
00328   if(HAL_SDRAM_Write_DMA(&sdramHandle, (uint32_t *)uwStartAddress, pData, uwDataSize) != HAL_OK)
00329       { return SDRAM_ERROR; }
00330   } else
00331   { return SDRAM_ERROR; }
00332 }
00333
00334 else
00335   { return SDRAM_OK; }
00336 }
00337
00338```
uint8_t BSP_SDRAM_Sendcmd(FMC_SDRAM_CommandTypeDef *SdramCmd) {
    if (HAL_SDRAM_SendCommand(&sdramHandle, SdramCmd, SDRAM_TIMEOUT) != HAL_OK) {
        return SDRAM_ERROR;
    } else {
        return SDRAM_OK;
    }
}

__weak void BSP_SDRAM_MspInit(SDRAM_HandleTypeDef *hsdram, void *Params) {
    static DMA_HandleTypeDef dma_handle;
    GPIO_InitTypeDef gpio_init_structure;
    /* Initialize SDRAM MSP. */
    /* @brief Initializes SDRAM MSP. */
    /* @param hsdram: SDRAM handle */
    /* @param Params */
    /* @retval None */
    /* @brief Sends command to the SDRAM bank. */
    /* @param SdramCmd: Pointer to SDRAM command structure */
    /* @retval SDRAM status */
/* Enable FMC clock */
__HAL_RCC_FMC_CLK_ENABLE();

/* Enable chosen DMAx clock */
__DMAx_CLK_ENABLE();

/* Enable GPIOs clock */
__HAL_RCC_GPIOD_CLK_ENABLE();
__HAL_RCC_GPIOE_CLK_ENABLE();
__HAL_RCC_GPIOF_CLK_ENABLE();
__HAL_RCC_GPIOG_CLK_ENABLE();
__HAL_RCC_GPIOH_CLK_ENABLE();
__HAL_RCC_GPIOI_CLK_ENABLE();

/* Common GPIO configuration */
gpio_init_structure.Mode = GPIO_MODE_AF_PP;
gpio_init_structure.Pull = GPIO_PULLUP;
gpio_init_structure.Speed = GPIO_SPEED_HIGH;
gpio_init_structure.Alternate = GPIO_AF12_FMC;

/* GPIOD configuration */
gpio_init_structure.Pin = GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_8 | GPIO_PIN_9 | GPIO_PIN_10 | GPIO_PIN_14 | GPIO_PIN_15;
HAL_GPIO_Init(GPIOD, &gpio_init_structure);

/* GPIOE configuration */
gpio_init_structure.Pin = GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_8 | GPIO_PIN_9 | GPIO_PIN_10 | GPIO_PIN_14 | GPIO_PIN_15;
PIO_PIN_1 | GPIO_PIN_7 | GPIO_PIN_8 | GPIO_PIN_9 |
00401 GPIO_PIN_10 |
GPIO_PIN_11 | GPIO_PIN_12 | GPIO_PIN_13 | GPIO_PIN_14 |
00402 GPIO_PIN_15;
00403
00404 HAL_GPIO_Init(GPIOE, &gpio_init_structure);
00405
00406 /* GPIOF configuration */
00407 gpio_init_structure.Pin = GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4 |
00408 GPIO_PIN_5 | GPIO_PIN_8 | GPIO_PIN_11 | GPIO_PIN_12 | GPIO_PIN_13 | GPIO_PIN_14 |
00409 GPIO_PIN_15;
00410
00411 HAL_GPIO_Init(GPIOF, &gpio_init_structure);
00412
00413 /* GPIOG configuration */
00414 gpio_init_structure.Pin = GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_4 |
00415 GPIO_PIN_5 | GPIO_PIN_8 | GPIO_PIN_15;
00416 HAL_GPIO_Init(GPIOG, &gpio_init_structure);
00417
00418 /* GPIOH configuration */
00419 gpio_init_structure.Pin = GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_5 | GPIO_PIN_8 | GPIO_PIN_9 |
00420 GPIO_PIN_10 | GPIO_PIN_11 | GPIO_PIN_12 | GPIO_PIN_13 | GPIO_PIN_14 |
00421 GPIO_PIN_15;
00422 HAL_GPIO_Init(GPIOH, &gpio_init_structure)
; 00423 00424  /* GPIOI configuration */ 00425  gpio_init_structure.Pin = GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6 | GPIO_PIN_7 | GPIO_PIN_9 | GPIO_PIN_10; 00426  HAL_GPIO_Init(GPIOI, &gpio_init_structure); 00427 00428  /* Configure common DMA parameters */ 00429 00430  dma_handle.Init.Channel = SDMA_M_DMAx_CHANNEL; 00431  dma_handle.Init.Direction = DMA_MEMORY_TO_MEMORY; 00432  dma_handle.InitPeriphInc = DMA_PINC_ENABLE; 00433  dma_handle.Init.MemInc = DMA_MINC_ENABLE; 00434  dma_handle.InitPeriphDataAlignment = DMA_PDATAALIGN_WORD; 00435  dma_handle.Init.MemDataAlignment = DMA_MDATAALIGN_WORD; 00436  dma_handle.Init.Mode = DMA_NORMAL; 00437  dma_handle.Init.Priority = DMA_PRIORITY_HIGH; 00438  dma_handle.Init.FIFOMode = DMA_FIFOMODE_DISABLE; 00439  dma_handle.Init.FIFOThreshold = DMA_FIFO_THRESHOLD_FULL; 00440  dma_handle.Init.MemBurst = DMA_MBURST_SINGLE; 00441  dma_handle.InitPeriphBurst = DMA_PBURST_SINGLE; 00442
dmai_handle.Instance = SDRAM_DMAx_STREAM;

/* Associate the DMA handle */
__HAL_LINKDMA(hsdram, hdma, dma_handle);

/* Deinitialize the stream for new transfer */
HAL_DMA_DeInit(&dma_handle);

/* Configure the DMA stream */
HAL_DMA_Init(&dma_handle);

/* NVIC configuration for DMA transfer complete interrupt */
HAL_NVIC_SetPriority(SDRAM_DMAx_IRQn, 0x0F, 0);
HAL_NVIC_EnableIRQ(SDRAM_DMAx_IRQn);
}

/**
 * @brief  DeInitializes SDRAM MSP.
 * @param  hsdram: SDRAM handle
 * @param  Params
 * @return None
 */
__weak void BSP_SDRAM_MspDeInit(SDRAM_HandleTypeDef *hsdram, void *Params)
{
    static DMA_HandleTypeDef dma_handle;
    /* Disable NVIC configuration for DMA interrupt */
    HAL_NVIC_DisableIRQ(SDRAM_DMAx_IRQHandler);
    /* Deinitialize the stream for new transfer */
    dma_handle.Instance = SDRAM_DMAx_STREAM;
HAL_DMA_DeInit(&dma_handle);

/* GPIO pins clock, FMC clock and DMA clock can be shut down in the applications by surcharging this __weak function */

*/

/**************************** (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
## STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BSP</strong></td>
<td></td>
<td></td>
<td>Modules</td>
</tr>
<tr>
<td>Modules</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
## STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### STM32F769I_DISCOVERY

<table>
<thead>
<tr>
<th>Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSP</td>
</tr>
</tbody>
</table>
## Modules

<table>
<thead>
<tr>
<th>Module Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F769I_DISCOVERY LOW LEVEL</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY AUDIO</td>
</tr>
<tr>
<td><strong>This file includes the low layer driver for wm8994 Audio Codec available on STM32F769I-DISCOVERY discoveryuation board (MB1225).</strong></td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY LCD</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY QSPI</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY QSPI</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY SDRAM</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY TS</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY EEPROM</td>
</tr>
<tr>
<td><strong>This file includes the I2C EEPROM driver of STM32F769I-DISCOVERY board.</strong></td>
</tr>
</tbody>
</table>

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directories</strong></td>
<td>Modules</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>STM32F769I DISCOVERY LCD Exported Macro</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>STM32F769I_DISCOVERY_LCD</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Modules

<table>
<thead>
<tr>
<th>STM32F769I DISCOVERY LCD Exported Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCD Exported Functions</td>
</tr>
</tbody>
</table>

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1
## STM32F769I-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Data Structures</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td>Modules</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### STM32F769I_DISCOVERY QSPI

STM32F769I_DISCOVERY
<table>
<thead>
<tr>
<th>Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F769I_DISCOVERY QSPI Private Variables</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY QSPI Private Functions</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY QSPI Exported Functions</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY_QSPI Exported Constants</td>
</tr>
<tr>
<td>STM32F769I_DISCOVERY_QSPI Exported Types</td>
</tr>
</tbody>
</table>

Generated on Fri Dec 30 2016 18:30:07 for STM32F769I-Discovery BSP User Manual by doxygen 1.7.6.1