STM32L0xx_Nucleo_32 BSP User Manual

Private Function Prototypes

NUCLEO 32

Generated on Mon Aug 28 2017 14:48:51 for STM32L0xx_Nucleo_32 BSP User Manual by doxygen 1.7.6.1

STM32L0xx_Nucleo_32 BSP User Manual

Exported Macros

Exported Constants

Generated on Mon Aug 28 2017 14:48:51 for STM32L0xx_Nucleo_32 BSP User Manual by doxygen 1.7.6.1

STM32L0xx_Nucleo_32 BSP User Manual

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

- _ -

- __STM32L0XX_NUCLEO_32_BSP_VERSION : **stm32l0xx_nucleo_32.c**
- __STM32L0XX_NUCLEO_32_BSP_VERSION_MAIN : **stm32l0xx_nucleo_32.c**
- __STM32L0XX_NUCLEO_32_BSP_VERSION_RC : **stm32l0xx_nucleo_32.c**
- __STM32L0XX_NUCLEO_32_BSP_VERSION_SUB1 : **stm32l0xx_nucleo_32.c**
- __STM32L0XX_NUCLEO_32_BSP_VERSION_SUB2 : **stm32l0xx_nucleo_32.c**

- a -

- ADCx_Init() : **stm32l0xx_nucleo_32.c**
- ADCx_MspInit() : **stm32l0xx_nucleo_32.c**

- b -

- BSP_GetVersion() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**
- BSP_I2C1 : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_CLK_DISABLE : **stm32l0xx_nucleo_32.h**

- BSP_I2C1_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_FORCE_RESET : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_GPIO_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_GPIO_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_GPIO_PORT : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_RELEASE_RESET : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_SCL_PIN : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_SCL_SDA_AF : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_SDA_PIN : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_TIMEOUT_MAX : **stm32l0xx_nucleo_32.h**
- BSP_JOY_GetState() : **stm32l0xx_nucleo_32.h** , **stm32l0xx_nucleo_32.c**
- BSP_JOY_Init() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**
- BSP_LED_Init() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**
- BSP_LED_Off() : **stm32l0xx_nucleo_32.h** , **stm32l0xx_nucleo_32.c**
- BSP_LED_On() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**
- BSP_LED_Toggle() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**

- h -

- heval_I2c1 : **stm32l0xx_nucleo_32.c**
- hnucleo_Adc : **stm32l0xx_nucleo_32.c**
- hnucleo_Spi : **stm32l0xx_nucleo_32.c**

- i -

- I2C1_Error() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**
- I2C1_Init() : **stm32l0xx_nucleo_32.h** , **stm32l0xx_nucleo_32.c**
- I2C1_IsDeviceReady() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**
- I2C1_MspInit() : **stm32l0xx_nucleo_32.h** , **stm32l0xx_nucleo_32.c**
- I2C1_Read() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**

- I2C1_ReadBuffer() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**
- I2C1_TIMING : **stm32l0xx_nucleo_32.h**
- I2C1_Write() : **stm32l0xx_nucleo_32.h** , **stm32l0xx_nucleo_32.c**
- I2C1_WriteBuffer() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**
- I2c1Timeout : **stm32l0xx_nucleo_32.c**

- j -

- JOY_DOWN : **stm32l0xx_nucleo_32.h**
- JOY_LEFT : **stm32l0xx_nucleo_32.h**
- JOY_NONE : **stm32l0xx_nucleo_32.h**
- JOY_RIGHT : **stm32l0xx_nucleo_32.h**
- JOY_SEL : **stm32l0xx_nucleo_32.h**
- JOY_UP : **stm32l0xx_nucleo_32.h**
- JOYState_TypeDef : **stm32l0xx_nucleo_32.h**

- l -

- LCD_CS_GPIO_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- LCD_CS_GPIO_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- LCD_CS_GPIO_PORT : **stm32l0xx_nucleo_32.h**
- LCD_CS_HIGH : **stm32l0xx_nucleo_32.h**
- LCD_CS_LOW : **stm32l0xx_nucleo_32.h**
- LCD_CS_PIN : **stm32l0xx_nucleo_32.h**
- LCD_DC_GPIO_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- LCD_DC_GPIO_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- LCD_DC_GPIO_PORT : **stm32l0xx_nucleo_32.h**
- LCD_DC_HIGH : **stm32l0xx_nucleo_32.h**
- LCD_DC_LOW : **stm32l0xx_nucleo_32.h**
- LCD_DC_PIN : **stm32l0xx_nucleo_32.h**
- LCD_Delay() : **stm32l0xx_nucleo_32.c**
- LCD_IO_Init() : **stm32l0xx_nucleo_32.c**
- LCD_IO_WriteData() : **stm32l0xx_nucleo_32.c**
- LCD_IO_WriteMultipleData() : **stm32l0xx_nucleo_32.c**
- LCD_IO_WriteReg() : **stm32l0xx_nucleo_32.c**
- LED3 : **stm32l0xx_nucleo_32.h**

- LED3_GPIO_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- LED3_GPIO_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- LED3_GPIO_PORT : **stm32l0xx_nucleo_32.h**
- LED3_PIN : **stm32l0xx_nucleo_32.h**
- LED_GREEN : **stm32l0xx_nucleo_32.h**
- LED_PIN : **stm32l0xx_nucleo_32.c**
- LED_PORT : **stm32l0xx_nucleo_32.c**
- Led_TypeDef : **stm32l0xx_nucleo_32.h**
- LEDn : **stm32l0xx_nucleo_32.h**
- LEDx_GPIO_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- LEDx_GPIO_CLK_ENABLE : **stm32l0xx_nucleo_32.h**

- n -

- NUCLEO_ADCx : **stm32l0xx_nucleo_32.h**
- NUCLEO_ADCx_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- NUCLEO_ADCx_GPIO_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- NUCLEO_ADCx_GPIO_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- NUCLEO_ADCx_GPIO_PIN : **stm32l0xx_nucleo_32.h**
- NUCLEO_ADCx_GPIO_PORT : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_MISO_MOSI_AF : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_MISO_MOSI_GPIO_PORT : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_MISO_PIN : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_MOSI_PIN : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_SCK_AF : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_SCK_GPIO_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_SCK_GPIO_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_SCK_GPIO_PORT : **stm32l0xx_nucleo_32.h**

- NUCLEO_SPIx_SCK_PIN : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_TIMEOUT_MAX : **stm32l0xx_nucleo_32.h**

- s -

- sConfig : **stm32l0xx_nucleo_32.c**
- SD_CS_GPIO_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- SD_CS_GPIO_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- SD_CS_GPIO_PORT : **stm32l0xx_nucleo_32.h**
- SD_CS_HIGH : **stm32l0xx_nucleo_32.h**
- SD_CS_LOW : **stm32l0xx_nucleo_32.h**
- SD_CS_PIN : **stm32l0xx_nucleo_32.h**
- SD_DUMMY_BYTE : **stm32l0xx_nucleo_32.c**
- SD_IO_Init() : **stm32l0xx_nucleo_32.c**
- SD_IO_ReadByte() : **stm32l0xx_nucleo_32.c**
- SD_IO_WaitResponse() : **stm32l0xx_nucleo_32.c**
- SD_IO_WriteByte() : **stm32l0xx_nucleo_32.c**
- SD_IO_WriteCmd() : **stm32l0xx_nucleo_32.c**
- SD_IO_WriteDummy() : **stm32l0xx_nucleo_32.c**
- SD_NO_RESPONSE_EXPECTED : **stm32l0xx_nucleo_32.c**
- SPIx_Error() : **stm32l0xx_nucleo_32.c**
- SPIx_Init() : **stm32l0xx_nucleo_32.c**
- SPIx_MspInit() : **stm32l0xx_nucleo_32.c**
- SPIx_Read() : **stm32l0xx_nucleo_32.c**
- SPIx_Write() : **stm32l0xx_nucleo_32.c**
- SpixTimeout : **stm32l0xx_nucleo_32.c**

STM32L0xx_Nucleo_32 BSP User Manual

- a -

- ADCx_Init() : **stm32l0xx_nucleo_32.c**
- ADCx_MspInit() : **stm32l0xx_nucleo_32.c**

- b -

- BSP_GetVersion() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**
- BSP_JOY_GetState() : **stm32l0xx_nucleo_32.h** , **stm32l0xx_nucleo_32.c**
- BSP_JOY_Init() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**
- BSP_LED_Init() : **stm32l0xx_nucleo_32.h** , **stm32l0xx_nucleo_32.c**
- BSP_LED_Off() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**
- BSP_LED_On() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**
- BSP_LED_Toggle() : **stm32l0xx_nucleo_32.h** , **stm32l0xx_nucleo_32.c**

- i -

- I2C1_Error() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**

- I2C1_Init() : **stm32l0xx_nucleo_32.h** , **stm32l0xx_nucleo_32.c**
- I2C1_IsDeviceReady() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**
- I2C1_MspInit() : **stm32l0xx_nucleo_32.h** , **stm32l0xx_nucleo_32.c**
- I2C1_Read() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**
- I2C1_ReadBuffer() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**
- I2C1_Write() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**
- I2C1_WriteBuffer() : **stm32l0xx_nucleo_32.c** , **stm32l0xx_nucleo_32.h**

- l -

- LCD_Delay() : **stm32l0xx_nucleo_32.c**
- LCD_IO_Init() : **stm32l0xx_nucleo_32.c**
- LCD_IO_WriteData() : **stm32l0xx_nucleo_32.c**
- LCD_IO_WriteMultipleData() : **stm32l0xx_nucleo_32.c**
- LCD_IO_WriteReg() : **stm32l0xx_nucleo_32.c**

- s -

- SD_IO_Init() : **stm32l0xx_nucleo_32.c**
- SD_IO_ReadByte() : **stm32l0xx_nucleo_32.c**
- SD_IO_WaitResponse() : **stm32l0xx_nucleo_32.c**
- SD_IO_WriteByte() : **stm32l0xx_nucleo_32.c**
- SD_IO_WriteCmd() : **stm32l0xx_nucleo_32.c**
- SD_IO_WriteDummy() : **stm32l0xx_nucleo_32.c**
- SPIx_Error() : **stm32l0xx_nucleo_32.c**
- SPIx_Init() : **stm32l0xx_nucleo_32.c**
- SPIx_MspInit() : **stm32l0xx_nucleo_32.c**
- SPIx_Read() : **stm32l0xx_nucleo_32.c**
- SPIx_Write() : **stm32l0xx_nucleo_32.c**

STM32L0xx_Nucleo_32 BSP User Manual

- heval_I2c1 : **stm32l0xx_nucleo_32.c**
- hnucleo_Adc : **stm32l0xx_nucleo_32.c**
- hnucleo_Spi : **stm32l0xx_nucleo_32.c**
- I2c1Timeout : **stm32l0xx_nucleo_32.c**
- LED_PIN : **stm32l0xx_nucleo_32.c**
- LED_PORT : **stm32l0xx_nucleo_32.c**
- sConfig : **stm32l0xx_nucleo_32.c**
- SpixTimeout : **stm32l0xx_nucleo_32.c**

STM32L0xx_Nucleo_32 BSP User Manual

- JOYState_TypeDef : **stm32l0xx_nucleo_32.h**
- Led_TypeDef : **stm32l0xx_nucleo_32.h**

Generated on Mon Aug 28 2017 14:48:51 for STM32L0xx_Nucleo_32 BSP User Manual by doxygen 1.7.6.1

STM32L0xx_Nucleo_32 BSP User Manual

- JOY_DOWN : **stm32l0xx_nucleo_32.h**
- JOY_LEFT : **stm32l0xx_nucleo_32.h**
- JOY_NONE : **stm32l0xx_nucleo_32.h**
- JOY_RIGHT : **stm32l0xx_nucleo_32.h**
- JOY_SEL : **stm32l0xx_nucleo_32.h**
- JOY_UP : **stm32l0xx_nucleo_32.h**
- LED3 : **stm32l0xx_nucleo_32.h**
- LED_GREEN : **stm32l0xx_nucleo_32.h**

STM32L0xx_Nucleo_32 BSP User Manual

- _ -

- __STM32L0XX_NUCLEO_32_BSP_VERSION : **stm32l0xx_nucleo_32.c**
- __STM32L0XX_NUCLEO_32_BSP_VERSION_MAIN : **stm32l0xx_nucleo_32.c**
- __STM32L0XX_NUCLEO_32_BSP_VERSION_RC : **stm32l0xx_nucleo_32.c**
- __STM32L0XX_NUCLEO_32_BSP_VERSION_SUB1 : **stm32l0xx_nucleo_32.c**
- __STM32L0XX_NUCLEO_32_BSP_VERSION_SUB2 : **stm32l0xx_nucleo_32.c**

- b -

- BSP_I2C1 : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_FORCE_RESET : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_GPIO_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_GPIO_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_GPIO_PORT : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_RELEASE_RESET : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_SCL_PIN : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_SCL_SDA_AF : **stm32l0xx_nucleo_32.h**

- BSP_I2C1_SDA_PIN : **stm32l0xx_nucleo_32.h**
- BSP_I2C1_TIMEOUT_MAX : **stm32l0xx_nucleo_32.h**

- i -

- I2C1_TIMING : **stm32l0xx_nucleo_32.h**

- l -

- LCD_CS_GPIO_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- LCD_CS_GPIO_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- LCD_CS_GPIO_PORT : **stm32l0xx_nucleo_32.h**
- LCD_CS_HIGH : **stm32l0xx_nucleo_32.h**
- LCD_CS_LOW : **stm32l0xx_nucleo_32.h**
- LCD_CS_PIN : **stm32l0xx_nucleo_32.h**
- LCD_DC_GPIO_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- LCD_DC_GPIO_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- LCD_DC_GPIO_PORT : **stm32l0xx_nucleo_32.h**
- LCD_DC_HIGH : **stm32l0xx_nucleo_32.h**
- LCD_DC_LOW : **stm32l0xx_nucleo_32.h**
- LCD_DC_PIN : **stm32l0xx_nucleo_32.h**
- LED3_GPIO_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- LED3_GPIO_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- LED3_GPIO_PORT : **stm32l0xx_nucleo_32.h**
- LED3_PIN : **stm32l0xx_nucleo_32.h**
- LEDn : **stm32l0xx_nucleo_32.h**
- LEDx_GPIO_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- LEDx_GPIO_CLK_ENABLE : **stm32l0xx_nucleo_32.h**

- n -

- NUCLEO_ADCx : **stm32l0xx_nucleo_32.h**
- NUCLEO_ADCx_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- NUCLEO_ADCx_GPIO_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- NUCLEO_ADCx_GPIO_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- NUCLEO_ADCx_GPIO_PIN : **stm32l0xx_nucleo_32.h**
- NUCLEO_ADCx_GPIO_PORT : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx : **stm32l0xx_nucleo_32.h**

- NUCLEO_SPIx_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_MISO_MOSI_AF : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_MISO_MOSI_GPIO_PORT : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_MISO_PIN : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_MOSI_PIN : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_SCK_AF : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_SCK_GPIO_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_SCK_GPIO_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_SCK_GPIO_PORT : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_SCK_PIN : **stm32l0xx_nucleo_32.h**
- NUCLEO_SPIx_TIMEOUT_MAX : **stm32l0xx_nucleo_32.h**

- s -

- SD_CS_GPIO_CLK_DISABLE : **stm32l0xx_nucleo_32.h**
- SD_CS_GPIO_CLK_ENABLE : **stm32l0xx_nucleo_32.h**
- SD_CS_GPIO_PORT : **stm32l0xx_nucleo_32.h**
- SD_CS_HIGH : **stm32l0xx_nucleo_32.h**
- SD_CS_LOW : **stm32l0xx_nucleo_32.h**
- SD_CS_PIN : **stm32l0xx_nucleo_32.h**
- SD_DUMMY_BYTE : **stm32l0xx_nucleo_32.c**
- SD_NO_RESPONSE_EXPECTED : **stm32l0xx_nucleo_32.c**

STM32L0xx_Nucleo_32 BSP User Manual

stm32l0xx_nucleo_32.c File Reference

This file provides set of firmware functions to manage: More...

```
#include "stm32l0xx_nucleo_32.h"
```

Go to the source code of this file.

Defines

#define	**__STM32L0XX_NUCLEO_32_BSP_VERSION_MAIN** (0x01) STM32L0XX NUCLEO BSP Driver version number.
#define	**__STM32L0XX_NUCLEO_32_BSP_VERSION_SUB1** (0x00)
#define	**__STM32L0XX_NUCLEO_32_BSP_VERSION_SUB2** (0x03)
#define	**__STM32L0XX_NUCLEO_32_BSP_VERSION_RC** (0x00)
#define	**__STM32L0XX_NUCLEO_32_BSP_VERSION**
#define	**SD_DUMMY_BYTE** 0xFF LINK SD Card.
#define	**SD_NO_RESPONSE_EXPECTED** 0x80

Functions

static void	**SPIx_Init** (void) Initializes SPI HAL.
static void	**SPIx_Write** (uint8_t Value) SPI Write a byte to device.
static uint32_t	**SPIx_Read** (void) SPI Read 4 bytes from device.
static void	**SPIx_Error** (void) SPI error treatment function.
static void	**SPIx_MspInit** (SPI_HandleTypeDef *hspi) Initializes SPI MSP.
void	**SD_IO_Init** (void) Initializes the SD Card and put it into StandBy State (Ready for data transfer).
HAL_StatusTypeDef	**SD_IO_WriteCmd** (uint8_t Cmd, uint32_t Arg, uint8_t Crc, uint8_t Response) Sends 5 bytes command to the SD card and get response.
HAL_StatusTypeDef	**SD_IO_WaitResponse** (uint8_t Response) Waits response from the SD card.
void	**SD_IO_WriteDummy** (void) Sends dummy byte with CS High.
void	**SD_IO_WriteByte** (uint8_t Data) Writes a byte on the SD.
uint8_t	**SD_IO_ReadByte** (void) Reads a byte from the SD.
void	**LCD_IO_Init** (void) Initializes the LCD.
void	**LCD_IO_WriteData** (uint8_t Data) Writes data to select the LCD register.
void	**LCD_IO_WriteMultipleData** (uint8_t *pData, uint32_t Size)

	Write register value.
void	**LCD_IO_WriteReg** (uint8_t LCDReg) Writes command to select the LCD register.
void	**LCD_Delay** (uint32_t Delay) Wait for loop in ms.
static void	**ADCx_Init** (void) Initializes ADC HAL.
static void	**ADCx_MspInit** (ADC_HandleTypeDef *hadc) Initializes ADC MSP.
uint32_t	**BSP_GetVersion** (void) This method returns the STM32L0XX NUCLEO BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LED GPIO.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**I2C1_Init** (void) I2C Bus initialization.
void	**I2C1_Write** (uint8_t Addr, uint8_t Reg, uint8_t Value) Writes a single data.
uint8_t	**I2C1_Read** (uint8_t Addr, uint8_t Reg) Reads a single data.
HAL_StatusTypeDef	**I2C1_ReadBuffer** (uint16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuffer, uint16_t Length) Reads multiple data on the BUS.
HAL_StatusTypeDef	**I2C1_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for

	communication.
HAL_StatusTypeDef	**I2C1_WriteBuffer** (uint16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuffer, uint16_t Length) Write a value in a register of the device through BUS.
void	**I2C1_Error** (void) Manages error callback by re-initializing I2C.
void	**I2C1_MspInit** (I2C_HandleTypeDef *hi2c) I2C MSP Initialization.
uint8_t	**BSP_JOY_Init** (void) Configures joystick available on adafruit 1.8" TFT shield managed through ADC to detect motion.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the Joystick key pressed.

Variables

GPIO_TypeDef *	**LED_PORT** [**LEDn**] = {**LED3_GPIO_PORT**}
const uint16_t	**LED_PIN** [**LEDn**] = {**LED3_PIN**}
uint32_t	**I2c1Timeout** = **BSP_I2C1_TIMEOUT_MAX** BUS variables.
I2C_HandleTypeDef	**heval_I2c1**
uint32_t	**SpixTimeout** = **NUCLEO_SPIx_TIMEOUT_MAX**
static SPI_HandleTypeDef	**hnucleo_Spi**
static ADC_HandleTypeDef	**hnucleo_Adc**
static ADC_ChannelConfTypeDef	**sConfig**

Detailed Description

This file provides set of firmware functions to manage:

Author:

MCD Application Team

- LEDs and push-button available on STM32L0XX-Nucleo Kit from STMicroelectronics

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 **stm32l0xx_nucleo_32.c**.

STM32L0xx_Nucleo_32 BSP User Manual

stm32l0xx_nucleo_32.h File Reference

This file contains definitions for: More...

```
#include "stm32l0xx_hal.h"
```

Go to the source code of this file.

Defines

```
#define LEDn 1
#define LED3_PIN GPIO_PIN_3
#define LED3_GPIO_PORT GPIOB
#define LED3_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_E
#define LED3_GPIO_CLK_DISABLE() __HAL_RCC_GPIOB_CLK_D
#define LEDx_GPIO_CLK_ENABLE(__INDEX__) do {LED3_GPIO_
#define LEDx_GPIO_CLK_DISABLE(__INDEX__) LED3_GPIO_CLK
#define BSP_I2C1 I2C1
#define BSP_I2C1_CLK_ENABLE() __HAL_RCC_I2C1_CLK_ENAB
#define BSP_I2C1_CLK_DISABLE() __HAL_RCC_I2C1_CLK_DISA
#define BSP_I2C1_FORCE_RESET() __HAL_RCC_I2C1_FORCE_R
#define BSP_I2C1_RELEASE_RESET() __HAL_RCC_I2C1_RELEA
#define BSP_I2C1_SCL_PIN GPIO_PIN_6 /* PB.6 add wire between
#define BSP_I2C1_SDA_PIN GPIO_PIN_7 /* PB.7 add wire between
#define BSP_I2C1_GPIO_PORT GPIOB /* GPIOB */
#define BSP_I2C1_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CL
#define BSP_I2C1_GPIO_CLK_DISABLE() __HAL_RCC_GPIOB_C
#define BSP_I2C1_SCL_SDA_AF GPIO_AF1_I2C1
#define BSP_I2C1_TIMEOUT_MAX 1000
#define I2C1_TIMING 0x009080B5
#define NUCLEO_SPIx SPI1
#define NUCLEO_SPIx_CLK_ENABLE() __HAL_RCC_SPI1_CLK_E
#define NUCLEO_SPIx_SCK_AF GPIO_AF0_SPI1
#define NUCLEO_SPIx_SCK_GPIO_PORT GPIOA
#define NUCLEO_SPIx_SCK_PIN GPIO_PIN_11
#define NUCLEO_SPIx_SCK_GPIO_CLK_ENABLE() __HAL_RCC_
#define NUCLEO_SPIx_SCK_GPIO_CLK_DISABLE() __HAL_RCC_
#define NUCLEO_SPIx_MISO_MOSI_AF GPIO_AF0_SPI1
#define NUCLEO_SPIx_MISO_MOSI_GPIO_PORT GPIOB
#define NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_ENABLE() __HAL
#define NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_DISABLE() __HA
```

#define	**NUCLEO_SPIx_MISO_PIN** GPIO_PIN_4
#define	**NUCLEO_SPIx_MOSI_PIN** GPIO_PIN_5
#define	**NUCLEO_SPIx_TIMEOUT_MAX** 1000
#define	**SD_CS_LOW**() HAL_GPIO_WritePin(**SD_CS_GPIO_PORT**, SD Control Lines management.
#define	**SD_CS_HIGH**() HAL_GPIO_WritePin(**SD_CS_GPIO_PORT**,
#define	**LCD_CS_LOW**() HAL_GPIO_WritePin(**LCD_CS_GPIO_POR** GPIO_PIN_RESET) LCD Control Lines management.
#define	**LCD_CS_HIGH**() HAL_GPIO_WritePin(**LCD_CS_GPIO_POR** GPIO_PIN_SET)
#define	**LCD_DC_LOW**() HAL_GPIO_WritePin(**LCD_DC_GPIO_POR** GPIO_PIN_RESET)
#define	**LCD_DC_HIGH**() HAL_GPIO_WritePin(**LCD_DC_GPIO_POR** GPIO_PIN_SET)
#define	**SD_CS_PIN** GPIO_PIN_5 SD Control Interface pins.
#define	**SD_CS_GPIO_PORT** GPIOB
#define	**SD_CS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_CLK_
#define	**SD_CS_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_CLK_
#define	**LCD_CS_PIN** GPIO_PIN_6 LCD Control Interface pins.
#define	**LCD_CS_GPIO_PORT** GPIOB
#define	**LCD_CS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_CLK
#define	**LCD_CS_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_CL
#define	**LCD_DC_PIN** GPIO_PIN_9 LCD Data/Command Interface pins.
#define	**LCD_DC_GPIO_PORT** GPIOA
#define	**LCD_DC_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOA_CLK
#define	**LCD_DC_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOA_CL
#define	**NUCLEO_ADCx** ADC1 ADC Interface pins used to detect motion of Joystick available
#define	**NUCLEO_ADCx_CLK_ENABLE**() __HAL_RCC_ADC1_CLK
#define	**NUCLEO_ADCx_GPIO_PORT** GPIOB

```
#define NUCLEO_ADCx_GPIO_PIN GPIO_PIN_0
#define NUCLEO_ADCx_GPIO_CLK_ENABLE() __HAL_RCC_GPIC
#define NUCLEO_ADCx_GPIO_CLK_DISABLE() __HAL_RCC_GPI
```

Enumerations

enum	**Led_TypeDef** { **LED3** = 0, **LED_GREEN** = LED3 }

enum	**JOYState_TypeDef** {
	JOY_NONE = 0, **JOY_SEL** = 1, **JOY_DOWN** = 2,
	JOY_LEFT = 3,
	JOY_RIGHT = 4, **JOY_UP** = 5
	}

Functions

void	**I2C1_Init** (void) I2C Bus initialization.
void	**I2C1_Error** (void) Manages error callback by re-initializing I2C.
void	**I2C1_MspInit** (I2C_HandleTypeDef *hi2c) I2C MSP Initialization.
void	**I2C1_Write** (uint8_t Addr, uint8_t Reg, uint8_t Value) Writes a single data.
uint8_t	**I2C1_Read** (uint8_t Addr, uint8_t Reg) Reads a single data.
HAL_StatusTypeDef	**I2C1_WriteBuffer** (uint16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuffer, uint16_t Length) Write a value in a register of the device through BUS.
HAL_StatusTypeDef	**I2C1_ReadBuffer** (uint16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuffer, uint16_t Length) Reads multiple data on the BUS.
HAL_StatusTypeDef	**I2C1_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.
uint32_t	**BSP_GetVersion** (void) This method returns the STM32L0XX NUCLEO BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LED GPIO.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.

void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
uint8_t	**BSP_JOY_Init** (void) Configures joystick available on adafruit 1.8" TFT shield managed through ADC to detect motion.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the Joystick key pressed.

Detailed Description

This file contains definitions for:

Author:

MCD Application Team

- LEDs and push-button available on STM32L0XX-Nucleo Kit from STMicroelectronics
- LCD, joystick and microSD available on Adafruit 1.8" TFT LCD shield (reference ID 802)

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 **stm32l0xx_nucleo_32.h**.

STM32L0xx_Nucleo_32 BSP User Manual

Modules

Here is a list of all modules:

STM32L0xx_Nucleo_32 BSP User Manual

File List

Here is a list of all files with brief descriptions:

stm32l0xx_nucleo_32.c [code]	This file provides set of firmware functions to manage:
stm32l0xx_nucleo_32.h [code]	This file contains definitions for:

STM32L0xx_Nucleo_32 BSP User Manual

Directories

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

- Firmware
  - Drivers
    - BSP
      - STM32L0xx_Nucleo_32

STM32L0xx_Nucleo_32 BSP User Manual

NUCLEO 32

BSP

This file provides set of firmware functions to manage Leds and push-button available on STM32L0XX-Nucleo Kit from STMicroelectronics.
More...

Modules

Detailed Description

This file provides set of firmware functions to manage Leds and push-button available on STM32L0XX-Nucleo Kit from STMicroelectronics.

STM32L0xx_Nucleo_32 BSP User Manual

Exported Constants

NUCLEO 32

Modules

- **Exported Macros**
- **Internal Functions**
- **Exported Functions**
- **STM32L0XX_NUCLEO_32_LED**

  Define for STM32L0XX_NUCLEO_32 board.

- **STM32L0XX_NUCLEO_32_BUS**
- **STM32L0XX_NUCLEO_32_COMPONENT**

STM32L0xx_Nucleo_32 BSP User Manual

Private Defines

NUCLEO 32

Defines

#define	**__STM32L0XX_NUCLEO_32_BSP_VERSION_MAIN** (0x01) STM32L0XX NUCLEO BSP Driver version number.
#define	**__STM32L0XX_NUCLEO_32_BSP_VERSION_SUB1** (0x00)
#define	**__STM32L0XX_NUCLEO_32_BSP_VERSION_SUB2** (0x03)
#define	**__STM32L0XX_NUCLEO_32_BSP_VERSION_RC** (0x00)
#define	**__STM32L0XX_NUCLEO_32_BSP_VERSION**
#define	**SD_DUMMY_BYTE** 0xFF LINK SD Card.
#define	**SD_NO_RESPONSE_EXPECTED** 0x80

Define Documentation

#define __STM32L0XX_NUCLEO_32_BSP_VERSION

Value:

```
((__STM32L0XX_NUCLEO_32_BSP_VERSION_MAIN << 24)\
                                             |(__
STM32L0XX_NUCLEO_32_BSP_VERSION_SUB1 << 16)\
                                             |(__
STM32L0XX_NUCLEO_32_BSP_VERSION_SUB2 << 8 )\
                                             |(__
STM32L0XX_NUCLEO_32_BSP_VERSION_RC))
```

Definition at line **62** of file **stm32l0xx_nucleo_32.c**.

Referenced by **BSP_GetVersion()**.

#define __STM32L0XX_NUCLEO_32_BSP_VERSION_MAIN (0x01)

STM32L0XX NUCLEO BSP Driver version number.

[31:24] main version

Definition at line **58** of file **stm32l0xx_nucleo_32.c**.

#define __STM32L0XX_NUCLEO_32_BSP_VERSION_RC (0x00)

[7:0] release candidate

Definition at line **61** of file **stm32l0xx_nucleo_32.c**.

#define __STM32L0XX_NUCLEO_32_BSP_VERSION_SUB1 (0x00)

[23:16] sub1 version

Definition at line **59** of file **stm32l0xx_nucleo_32.c**.

#define __STM32L0XX_NUCLEO_32_BSP_VERSION_SUB2 (0x03)

[15:8] sub2 version

Definition at line **60** of file **stm32l0xx_nucleo_32.c**.

#define SD_DUMMY_BYTE 0xFF

LINK SD Card.

Definition at line **70** of file **stm32l0xx_nucleo_32.c**.

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

#define SD_NO_RESPONSE_EXPECTED 0x80

Definition at line **71** of file **stm32l0xx_nucleo_32.c**.

Referenced by **SD_IO_WriteCmd()**.

STM32L0xx_Nucleo_32 BSP User Manual

Private Functions

NUCLEO 32

Functions

uint32_t	**BSP_GetVersion** (void) This method returns the STM32L0XX NUCLEO BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LED GPIO.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**I2C1_Init** (void) I2C Bus initialization.
void	**I2C1_Write** (uint8_t Addr, uint8_t Reg, uint8_t Value) Writes a single data.
uint8_t	**I2C1_Read** (uint8_t Addr, uint8_t Reg) Reads a single data.
HAL_StatusTypeDef	**I2C1_ReadBuffer** (uint16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuffer, uint16_t Length) Reads multiple data on the BUS.
HAL_StatusTypeDef	**I2C1_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.
HAL_StatusTypeDef	**I2C1_WriteBuffer** (uint16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuffer, uint16_t Length) Write a value in a register of the device through BUS.

void	**I2C1_Error** (void) Manages error callback by re-initializing I2C.
void	**I2C1_MspInit** (I2C_HandleTypeDef *hi2c) I2C MSP Initialization.
uint8_t	**BSP_JOY_Init** (void) Configures joystick available on adafruit 1.8" TFT shield managed through ADC to detect motion.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the Joystick key pressed.
static void	**SPIx_MspInit** (SPI_HandleTypeDef *hspi) Initializes SPI MSP.
static void	**SPIx_Init** (void) Initializes SPI HAL.
static uint32_t	**SPIx_Read** (void) SPI Read 4 bytes from device.
static void	**SPIx_Write** (uint8_t Value) SPI Write a byte to device.
static void	**SPIx_Error** (void) SPI error treatment function.
void	**SD_IO_Init** (void) Initializes the SD Card and put it into StandBy State (Ready for data transfer).
void	**SD_IO_WriteByte** (uint8_t Data) Writes a byte on the SD.
uint8_t	**SD_IO_ReadByte** (void) Reads a byte from the SD.
HAL_StatusTypeDef	**SD_IO_WriteCmd** (uint8_t Cmd, uint32_t Arg, uint8_t Crc, uint8_t Response) Sends 5 bytes command to the SD card and get response.
HAL_StatusTypeDef	**SD_IO_WaitResponse** (uint8_t Response) Waits response from the SD card.
void	**SD_IO_WriteDummy** (void)

	Sends dummy byte with CS High.
void	**LCD_IO_Init** (void) Initializes the LCD.
void	**LCD_IO_WriteReg** (uint8_t LCDReg) Writes command to select the LCD register.
void	**LCD_IO_WriteData** (uint8_t Data) Writes data to select the LCD register.
void	**LCD_IO_WriteMultipleData** (uint8_t *pData, uint32_t Size) Write register value.
void	**LCD_Delay** (uint32_t Delay) Wait for loop in ms.
static void	**ADCx_MspInit** (ADC_HandleTypeDef *hadc) Initializes ADC MSP.
static void	**ADCx_Init** (void) Initializes ADC HAL.

Function Documentation

static void ADCx_Init (void) [static]

Initializes ADC HAL.

Return values:
None

Definition at line **836** of file **stm32l0xx_nucleo_32.c**.

References **ADCx_MspInit()**, **hnucleo_Adc**, and **NUCLEO_ADCx**.

Referenced by **BSP_JOY_Init()**.

static void ADCx_MspInit (ADC_HandleTypeDef * hadc) [static]

Initializes ADC MSP.

Parameters:
hadc,: ADC peripheral

Return values:
None

Definition at line **814** of file **stm32l0xx_nucleo_32.c**.

References **NUCLEO_ADCx_CLK_ENABLE**, **NUCLEO_ADCx_GPIO_CLK_ENABLE**, **NUCLEO_ADCx_GPIO_PIN**, and **NUCLEO_ADCx_GPIO_PORT**.

Referenced by **ADCx_Init()**.

uint32_t BSP_GetVersion (void)

This method returns the STM32L0XX NUCLEO BSP Driver revision.

Return values:

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

Definition at line **148** of file **stm32l0xx_nucleo_32.c**.

References **__STM32L0XX_NUCLEO_32_BSP_VERSION**.

JOYState_TypeDef BSP_JOY_GetState (void)

Returns the Joystick key pressed.

Note:

To know which Joystick key is pressed we need to detect the voltage level on each key output

- None : 3.3 V / 4095
- SEL : 1.055 V / 1308
- DOWN : 0.71 V / 88
- LEFT : 3.0 V / 3720
- RIGHT : 0.595 V / 737
- UP : 1.65 V / 2046

Return values:

JOYState_TypeDef,: Code of the Joystick key pressed.

Definition at line **896** of file **stm32l0xx_nucleo_32.c**.

References **hnucleo_Adc**, **JOY_DOWN**, **JOY_LEFT**, **JOY_NONE**, **JOY_RIGHT**, **JOY_SEL**, and **JOY_UP**.

uint8_t BSP_JOY_Init (void)

Configures joystick available on adafruit 1.8" TFT shield managed through ADC to detect motion.

Return values:

Joystickstatus (0=> success, 1=> fail)

Definition at line **867** of file **stm32l0xx_nucleo_32.c**.

References **ADCx_Init()**, **hnucleo_Adc**, and **sConfig**.

void BSP_LED_Init (Led_TypeDef Led)

Configures LED GPIO.

Parameters:

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

- LED3

Return values:

None

Definition at line **160** of file **stm32l0xx_nucleo_32.c**.

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

void BSP_LED_Off (Led_TypeDef Led)

Turns selected LED Off.

Parameters:

Led,: Specifies the Led to be set off. This parameter can be one of following parameters:

- LED3

Return values:

None

Definition at line **196** of file **stm32l0xx_nucleo_32.c**.

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

void BSP_LED_On (Led_TypeDef Led)

Turns selected LED On.

Parameters:

Led,: Specifies the Led to be set on. This parameter can be one of following parameters:

- LED3

Return values:

None

Definition at line **184** of file **stm32l0xx_nucleo_32.c**.

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

void BSP_LED_Toggle (Led_TypeDef Led)

Toggles the selected LED.

Parameters:

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

- LED3

Return values:

None

Definition at line **208** of file **stm32l0xx_nucleo_32.c**.

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

void I2C1_Error (void)

Manages error callback by re-initializing I2C.

Return values:
None

Definition at line **352** of file **stm32l0xx_nucleo_32.c**.

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

Referenced by **I2C1_Read()**, **I2C1_ReadBuffer()**, **I2C1_Write()**, and **I2C1_WriteBuffer()**.

void I2C1_Init (void)

I2C Bus initialization.

Return values:
None

Definition at line **223** of file **stm32l0xx_nucleo_32.c**.

References **BSP_I2C1**, **heval_I2c1**, **I2C1_MspInit()**, and **I2C1_TIMING**.

Referenced by **I2C1_Error()**.

HAL_StatusTypeDef I2C1_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 **319** of file **stm32l0xx_nucleo_32.c**.

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

void I2C1_MspInit (I2C_HandleTypeDef * hi2c)

I2C MSP Initialization.

Parameters:

hi2c,: I2C handle

Return values:

None

Definition at line **366** of file **stm32l0xx_nucleo_32.c**.

References **BSP_I2C1_CLK_ENABLE**, **BSP_I2C1_FORCE_RESET**, **BSP_I2C1_GPIO_CLK_ENABLE**, **BSP_I2C1_GPIO_PORT**, **BSP_I2C1_RELEASE_RESET**, **BSP_I2C1_SCL_PIN**, **BSP_I2C1_SCL_SDA_AF**, and **BSP_I2C1_SDA_PIN**.

Referenced by **I2C1_Init()**.

uint8_t I2C1_Read (uint8_t Addr,
uint8_t Reg

)

Reads a single data.

Parameters:

Addr,: I2C address

Reg,: Register address

Return values:

Read data

Definition at line **270** of file **stm32l0xx_nucleo_32.c**.

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

HAL_StatusTypeDef I2C1_ReadBuffer (uint16_t Addr,
uint8_t Reg,
uint16_t RegSize,
uint8_t * pBuffer,
uint16_t Length
)

Reads multiple data on the BUS.

Parameters:

Addr : I2C Address

Reg : Reg Address

RegSize : The target register size (can be 8BIT or 16BIT)

pBuffer : pointer to read data buffer

Length : length of the data

Return values:

0 if no problems to read multiple data

Definition at line 297 of file stm32l0xx_nucleo_32.c.

References heval_I2c1, I2C1_Error(), and I2c1Timeout.

```
void I2C1_Write ( uint8_t  Addr,
                  uint8_t  Reg,
                  uint8_t  Value
                )
```

Writes a single data.

Parameters:

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

Return values:

- **None**

Definition at line 250 of file stm32l0xx_nucleo_32.c.

References heval_I2c1, and I2C1_Error().

```
HAL_StatusTypeDef I2C1_WriteBuffer ( uint16_t  Addr,
                                     uint8_t   Reg,
                                     uint16_t  RegSize,
                                     uint8_t *  pBuffer,
                                     uint16_t  Length
                                   )
```

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

Parameters:

- **Addr,:** Device address on BUS Bus.

Reg,: The target register address to write
RegSize,: The target register size (can be 8BIT or 16BIT)
pBuffer,: The target register value to be written
Length,: buffer size to be written

Return values:
None

Definition at line **333** of file **stm32l0xx_nucleo_32.c**.

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

void LCD_Delay (uint32_t Delay)

Wait for loop in ms.

Parameters:
Delay in ms.

Return values:
None

Definition at line **801** of file **stm32l0xx_nucleo_32.c**.

void LCD_IO_Init (void)

Initializes the LCD.

Return values:
None

Definition at line **678** of file **stm32l0xx_nucleo_32.c**.

References **LCD_CS_GPIO_CLK_ENABLE**, **LCD_CS_GPIO_PORT**, **LCD_CS_HIGH**, **LCD_CS_PIN**, **LCD_DC_GPIO_CLK_ENABLE**,

LCD_DC_GPIO_PORT, LCD_DC_PIN, and SPIx_Init().

void LCD_IO_WriteData (uint8_t Data)

Writes data to select the LCD register.

This function must be used after st7735_WriteReg() function

Parameters:

Data,: data to write to the selected register.

Return values:

None

Definition at line 730 of file stm32l0xx_nucleo_32.c.

References LCD_CS_HIGH, LCD_CS_LOW, LCD_DC_HIGH, and SPIx_Write().

void LCD_IO_WriteMultipleData (uint8_t * pData, uint32_t Size)

Write register value.

Parameters:

pData Pointer on the register value

Size Size of byte to transmit to the register

Return values:

None

Definition at line 751 of file stm32l0xx_nucleo_32.c.

References hnucleo_Spi, LCD_CS_HIGH, LCD_CS_LOW,

LCD_DC_HIGH, and SPIx_Write().

void LCD_IO_WriteReg (uint8_t LCDReg)

Writes command to select the LCD register.

Parameters:

LCDReg,: Address of the selected register.

Return values:

None

Definition at line 709 of file stm32l0xx_nucleo_32.c.

References LCD_CS_HIGH, LCD_CS_LOW, LCD_DC_LOW, and SPIx_Write().

void SD_IO_Init (void)

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

Return values:

None

Definition at line 538 of file stm32l0xx_nucleo_32.c.

References SD_CS_GPIO_CLK_ENABLE, SD_CS_GPIO_PORT, SD_CS_HIGH, SD_CS_PIN, SD_DUMMY_BYTE, SD_IO_WriteByte(), and SPIx_Init().

uint8_t SD_IO_ReadByte (void)

Reads a byte from the SD.

Return values:

The received byte.

Definition at line **584** of file **stm32l0xx_nucleo_32.c**.

References **SPIx_Read()**.

Referenced by **SD_IO_WaitResponse()**.

HAL_StatusTypeDef SD_IO_WaitResponse (uint8_t Response)

Waits response from the SD card.

Parameters:

Response,: Expected response from the SD card

Return values:

HAL_StatusTypeDef HAL Status

Definition at line **638** of file **stm32l0xx_nucleo_32.c**.

References **SD_IO_ReadByte()**.

Referenced by **SD_IO_WriteCmd()**.

void SD_IO_WriteByte (uint8_t Data)

Writes a byte on the SD.

Parameters:

Data,: byte to send.

Return values:

None

Definition at line 574 of file stm32l0xx_nucleo_32.c.

References SPIx_Write().

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

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

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

Parameters:

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

Return values:

HAL_StatusTypeDef	HAL Status

Definition at line 603 of file stm32l0xx_nucleo_32.c.

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

void SD_IO_WriteDummy (void)

Sends dummy byte with CS High.

Return values:

None

Definition at line **664** of file **stm32l0xx_nucleo_32.c**.

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

static void SPIx_Error (void) [static]

SPI error treatment function.

Return values:
None

Definition at line **519** of file **stm32l0xx_nucleo_32.c**.

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

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

static void SPIx_Init (void) [static]

Initializes SPI HAL.

Return values:
None

Definition at line **443** of file **stm32l0xx_nucleo_32.c**.

References **hnucleo_Spi**, **NUCLEO_SPIx**, and **SPIx_MspInit()**.

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

static void SPIx_MspInit (SPI_HandleTypeDef * hspi) [static]

Initializes SPI MSP.

Parameters:

hspi,: SPI handle

Return values:

None

Definition at line **408** of file **stm32l0xx_nucleo_32.c**.

References **NUCLEO_SPIx_CLK_ENABLE**, **NUCLEO_SPIx_MISO_MOSI_AF**, **NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_ENABLE**, **NUCLEO_SPIx_MISO_MOSI_GPIO_PORT**, **NUCLEO_SPIx_MISO_PIN**, **NUCLEO_SPIx_MOSI_PIN**, **NUCLEO_SPIx_SCK_AF**, **NUCLEO_SPIx_SCK_GPIO_CLK_ENABLE**, **NUCLEO_SPIx_SCK_GPIO_PORT**, and **NUCLEO_SPIx_SCK_PIN**.

Referenced by **SPIx_Init()**.

static uint32_t SPIx_Read (void) [static]

SPI Read 4 bytes from device.

Return values:

Read data

Definition at line **478** of file **stm32l0xx_nucleo_32.c**.

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

Referenced by **SD_IO_ReadByte()**.

static void SPIx_Write (uint8_t Value) [static]

SPI Write a byte to device.

Parameters:

Value,: value to be written

Return values:

None

Definition at line **501** of file **stm32l0xx_nucleo_32.c**.

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

Referenced by **LCD_IO_WriteData()**, **LCD_IO_WriteMultipleData()**, **LCD_IO_WriteReg()**, and **SD_IO_WriteByte()**.

STM32L0xx_Nucleo_32 BSP User Manual

Exported Functions

Exported Constants

Functions

uint32_t	**BSP_GetVersion** (void) This method returns the STM32L0XX NUCLEO BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LED GPIO.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
uint8_t	**BSP_JOY_Init** (void) Configures joystick available on adafruit 1.8" TFT shield managed through ADC to detect motion.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the Joystick key pressed.

Function Documentation

uint32_t BSP_GetVersion (void)

This method returns the STM32L0XX NUCLEO BSP Driver revision.

Return values:

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

Definition at line **148** of file **stm32l0xx_nucleo_32.c**.

References **__STM32L0XX_NUCLEO_32_BSP_VERSION**.

JOYState_TypeDef BSP_JOY_GetState (void)

Returns the Joystick key pressed.

Note:

To know which Joystick key is pressed we need to detect the voltage level on each key output

- None : 3.3 V / 4095
- SEL : 1.055 V / 1308
- DOWN : 0.71 V / 88
- LEFT : 3.0 V / 3720
- RIGHT : 0.595 V / 737
- UP : 1.65 V / 2046

Return values:

JOYState_TypeDef,: Code of the Joystick key pressed.

Definition at line **896** of file **stm32l0xx_nucleo_32.c**.

References **hnucleo_Adc**, **JOY_DOWN**, **JOY_LEFT**, **JOY_NONE**, **JOY_RIGHT**, **JOY_SEL**, and **JOY_UP**.

uint8_t BSP_JOY_Init (void)

Configures joystick available on adafruit 1.8" TFT shield managed through ADC to detect motion.

Return values:

Joystickstatus (0=> success, 1=> fail)

Definition at line **867** of file **stm32l0xx_nucleo_32.c**.

References **ADCx_Init()**, **hnucleo_Adc**, and **sConfig**.

void BSP_LED_Init (Led_TypeDef Led)

Configures LED GPIO.

Parameters:

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

- LED3

Return values:

None

Definition at line **160** of file **stm32l0xx_nucleo_32.c**.

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

void BSP_LED_Off (Led_TypeDef Led)

Turns selected LED Off.

Parameters:

Led,: Specifies the Led to be set off. This parameter can be

one of following parameters:
- LED3

Return values:
None

Definition at line **196** of file **stm32l0xx_nucleo_32.c**.

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

void BSP_LED_On (Led_TypeDef Led)

Turns selected LED On.

Parameters:
Led,: Specifies the Led to be set on. This parameter can be one of following parameters:
- LED3

Return values:
None

Definition at line **184** of file **stm32l0xx_nucleo_32.c**.

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

void BSP_LED_Toggle (Led_TypeDef Led)

Toggles the selected LED.

Parameters:
Led,: Specifies the Led to be toggled. This parameter can be one of following parameters:
- LED3

Return values:

None

Definition at line **208** of file **stm32l0xx_nucleo_32.c**.

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

STM32L0xx_Nucleo_32 BSP User Manual

STM32L0XX_NUCLEO_32_BUS

Exported Constants

Defines

#define	**BSP_I2C1** I2C1
#define	**BSP_I2C1_CLK_ENABLE**() __HAL_RCC_I2C1_CLK_ENAB
#define	**BSP_I2C1_CLK_DISABLE**() __HAL_RCC_I2C1_CLK_DISA
#define	**BSP_I2C1_FORCE_RESET**() __HAL_RCC_I2C1_FORCE_R
#define	**BSP_I2C1_RELEASE_RESET**() __HAL_RCC_I2C1_RELEA
#define	**BSP_I2C1_SCL_PIN** GPIO_PIN_6 /* PB.6 add wire between
#define	**BSP_I2C1_SDA_PIN** GPIO_PIN_7 /* PB.7 add wire between
#define	**BSP_I2C1_GPIO_PORT** GPIOB /* GPIOB */
#define	**BSP_I2C1_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_CL
#define	**BSP_I2C1_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_C
#define	**BSP_I2C1_SCL_SDA_AF** GPIO_AF1_I2C1
#define	**BSP_I2C1_TIMEOUT_MAX** 1000
#define	**I2C1_TIMING** 0x009080B5
#define	**NUCLEO_SPIx** SPI1
#define	**NUCLEO_SPIx_CLK_ENABLE**() __HAL_RCC_SPI1_CLK_E
#define	**NUCLEO_SPIx_SCK_AF** GPIO_AF0_SPI1
#define	**NUCLEO_SPIx_SCK_GPIO_PORT** GPIOA
#define	**NUCLEO_SPIx_SCK_PIN** GPIO_PIN_11
#define	**NUCLEO_SPIx_SCK_GPIO_CLK_ENABLE**() __HAL_RCC_
#define	**NUCLEO_SPIx_SCK_GPIO_CLK_DISABLE**() __HAL_RCC_
#define	**NUCLEO_SPIx_MISO_MOSI_AF** GPIO_AF0_SPI1
#define	**NUCLEO_SPIx_MISO_MOSI_GPIO_PORT** GPIOB
#define	**NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_ENABLE**() __HAL
#define	**NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_DISABLE**() __HA
#define	**NUCLEO_SPIx_MISO_PIN** GPIO_PIN_4
#define	**NUCLEO_SPIx_MOSI_PIN** GPIO_PIN_5
#define	**NUCLEO_SPIx_TIMEOUT_MAX** 1000

Define Documentation

#define BSP_I2C1 I2C1

Definition at line 119 of file stm32l0xx_nucleo_32.h.

Referenced by I2C1_Init().

#define BSP_I2C1_CLK_DISABLE ( ) __HAL_RCC_I2C1_CLK_DIS

Definition at line 121 of file stm32l0xx_nucleo_32.h.

#define BSP_I2C1_CLK_ENABLE ( ) __HAL_RCC_I2C1_CLK_ENA

Definition at line 120 of file stm32l0xx_nucleo_32.h.

Referenced by I2C1_MspInit().

#define BSP_I2C1_FORCE_RESET ( ) __HAL_RCC_I2C1_FORCE_

Definition at line 122 of file stm32l0xx_nucleo_32.h.

Referenced by I2C1_MspInit().

#define BSP_I2C1_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOB_

Definition at line 130 of file stm32l0xx_nucleo_32.h.

#define BSP_I2C1_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOB_(

Definition at line 129 of file stm32l0xx_nucleo_32.h.

Referenced by I2C1_MspInit().

#define BSP_I2C1_GPIO_PORT GPIOB /* GPIOB */

Definition at line 128 of file stm32l0xx_nucleo_32.h.

Referenced by I2C1_MspInit().

#define BSP_I2C1_RELEASE_RESET ( ) __HAL_RCC_I2C1_RELE

Definition at line 123 of file stm32l0xx_nucleo_32.h.

Referenced by I2C1_MspInit().

#define BSP_I2C1_SCL_PIN GPIO_PIN_6 /* PB.6 add wire betwee

Definition at line 125 of file stm32l0xx_nucleo_32.h.

Referenced by I2C1_MspInit().

#define BSP_I2C1_SCL_SDA_AF GPIO_AF1_I2C1

Definition at line 131 of file stm32l0xx_nucleo_32.h.

Referenced by I2C1_MspInit().

#define BSP_I2C1_SDA_PIN GPIO_PIN_7 /* PB.7 add wire betwee

Definition at line 126 of file stm32l0xx_nucleo_32.h.

Referenced by I2C1_MspInit().

#define BSP_I2C1_TIMEOUT_MAX 1000

Definition at line 138 of file stm32l0xx_nucleo_32.h.

#define I2C1_TIMING 0x009080B5

Definition at line 142 of file stm32l0xx_nucleo_32.h.

Referenced by I2C1_Init().

#define NUCLEO_SPIx SPI1

Definition at line 148 of file stm32l0xx_nucleo_32.h.

Referenced by SPIx_Init().

#define NUCLEO_SPIx_CLK_ENABLE ( ) __HAL_RCC_SPI1_CLK

Definition at line 149 of file stm32l0xx_nucleo_32.h.

Referenced by SPIx_MspInit().

#define NUCLEO_SPIx_MISO_MOSI_AF GPIO_AF0_SPI1

Definition at line 157 of file stm32l0xx_nucleo_32.h.

Referenced by SPIx_MspInit().

#define NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_DISABLE ( ) __H

Definition at line 160 of file stm32l0xx_nucleo_32.h.

#define NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_ENABLE ( ) __HA

Definition at line 159 of file stm32l0xx_nucleo_32.h.

Referenced by SPIx_MspInit().

#define NUCLEO_SPIx_MISO_MOSI_GPIO_PORT GPIOB

Definition at line 158 of file stm32l0xx_nucleo_32.h.

Referenced by SPIx_MspInit().

#define NUCLEO_SPIx_MISO_PIN GPIO_PIN_4

Definition at line 161 of file stm32l0xx_nucleo_32.h.

Referenced by SPIx_MspInit().

#define NUCLEO_SPIx_MOSI_PIN GPIO_PIN_5

Definition at line 162 of file stm32l0xx_nucleo_32.h.

Referenced by SPIx_MspInit().

#define NUCLEO_SPIx_SCK_AF GPIO_AF0_SPI1

Definition at line 151 of file stm32l0xx_nucleo_32.h.

Referenced by SPIx_MspInit().

#define NUCLEO_SPIx_SCK_GPIO_CLK_DISABLE ( ) __HAL_RC

Definition at line **155** of file **stm32l0xx_nucleo_32.h**.

#define NUCLEO_SPIx_SCK_GPIO_CLK_ENABLE ( ) __HAL_RCC

Definition at line **154** of file **stm32l0xx_nucleo_32.h**.

Referenced by **SPIx_MspInit()**.

#define NUCLEO_SPIx_SCK_GPIO_PORT GPIOA

Definition at line **152** of file **stm32l0xx_nucleo_32.h**.

Referenced by **SPIx_MspInit()**.

#define NUCLEO_SPIx_SCK_PIN GPIO_PIN_11

Definition at line **153** of file **stm32l0xx_nucleo_32.h**.

Referenced by **SPIx_MspInit()**.

#define NUCLEO_SPIx_TIMEOUT_MAX 1000

Definition at line **168** of file **stm32l0xx_nucleo_32.h**.

STM32L0xx_Nucleo_32 BSP User Manual

Private Variables

NUCLEO 32

Variables

GPIO_TypeDef *	**LED_PORT** [**LEDn**] = {**LED3_GPIO_PORT**}
const uint16_t	**LED_PIN** [**LEDn**] = {**LED3_PIN**}
uint32_t	**I2c1Timeout** = **BSP_I2C1_TIMEOUT_MAX** BUS variables.
I2C_HandleTypeDef	**heval_I2c1**
uint32_t	**SpixTimeout** = **NUCLEO_SPIx_TIMEOUT_MAX**
static SPI_HandleTypeDef	**hnucleo_Spi**
static ADC_HandleTypeDef	**hnucleo_Adc**
static ADC_ChannelConfTypeDef	**sConfig**

Variable Documentation

I2C_HandleTypeDef heval_I2c1

Definition at line 88 of file stm32l0xx_nucleo_32.c.

Referenced by I2C1_Error(), I2C1_Init(), I2C1_IsDeviceReady(), I2C1_Read(), I2C1_ReadBuffer(), I2C1_Write(), and I2C1_WriteBuffer().

ADC_HandleTypeDef hnucleo_Adc [static]

Definition at line 97 of file stm32l0xx_nucleo_32.c.

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

SPI_HandleTypeDef hnucleo_Spi [static]

Definition at line 93 of file stm32l0xx_nucleo_32.c.

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

uint32_t I2c1Timeout = BSP_I2C1_TIMEOUT_MAX

BUS variables.

Definition at line 87 of file stm32l0xx_nucleo_32.c.

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

const uint16_t LED_PIN[LEDn] = {LED3_PIN}

Definition at line **81** of file **stm32l0xx_nucleo_32.c**.

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

GPIO_TypeDef* LED_PORT[LEDn] = {LED3_GPIO_PORT}

Definition at line **80** of file **stm32l0xx_nucleo_32.c**.

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

ADC_ChannelConfTypeDef sConfig `[static]`

Definition at line **99** of file **stm32l0xx_nucleo_32.c**.

Referenced by **BSP_JOY_Init()**.

uint32_t SpixTimeout = NUCLEO_SPIx_TIMEOUT_MAX

Definition at line **92** of file **stm32l0xx_nucleo_32.c**.

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

STM32L0xx_Nucleo_32 BSP User Manual

Internal Functions

Exported Constants

Functions

void	**I2C1_Init** (void) I2C Bus initialization.
void	**I2C1_Error** (void) Manages error callback by re-initializing I2C.
void	**I2C1_MspInit** (I2C_HandleTypeDef *hi2c) I2C MSP Initialization.
void	**I2C1_Write** (uint8_t Addr, uint8_t Reg, uint8_t Value) Writes a single data.
uint8_t	**I2C1_Read** (uint8_t Addr, uint8_t Reg) Reads a single data.
HAL_StatusTypeDef	**I2C1_WriteBuffer** (uint16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuffer, uint16_t Length) Write a value in a register of the device through BUS.
HAL_StatusTypeDef	**I2C1_ReadBuffer** (uint16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuffer, uint16_t Length) Reads multiple data on the BUS.
HAL_StatusTypeDef	**I2C1_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.

Function Documentation

void I2C1_Error (void)

Manages error callback by re-initializing I2C.

Return values:
None

Definition at line **352** of file **stm32l0xx_nucleo_32.c**.

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

Referenced by **I2C1_Read()**, **I2C1_ReadBuffer()**, **I2C1_Write()**, and **I2C1_WriteBuffer()**.

void I2C1_Init (void)

I2C Bus initialization.

Return values:
None

Definition at line **223** of file **stm32l0xx_nucleo_32.c**.

References **BSP_I2C1**, **heval_I2c1**, **I2C1_MspInit()**, and **I2C1_TIMING**.

Referenced by **I2C1_Error()**.

HAL_StatusTypeDef I2C1_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 **319** of file **stm32l0xx_nucleo_32.c**.

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

void I2C1_MspInit (I2C_HandleTypeDef * hi2c)

I2C MSP Initialization.

Parameters:

hi2c,: I2C handle

Return values:

None

Definition at line **366** of file **stm32l0xx_nucleo_32.c**.

References **BSP_I2C1_CLK_ENABLE**, **BSP_I2C1_FORCE_RESET**, **BSP_I2C1_GPIO_CLK_ENABLE**, **BSP_I2C1_GPIO_PORT**, **BSP_I2C1_RELEASE_RESET**, **BSP_I2C1_SCL_PIN**, **BSP_I2C1_SCL_SDA_AF**, and **BSP_I2C1_SDA_PIN**.

Referenced by **I2C1_Init()**.

uint8_t I2C1_Read (uint8_t Addr,
uint8_t Reg
)

Reads a single data.

Parameters:

Addr,: I2C address

Reg,: Register address

Return values:

Read data

Definition at line **270** of file **stm32l0xx_nucleo_32.c**.

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

HAL_StatusTypeDef I2C1_ReadBuffer (uint16_t Addr,
uint8_t Reg,
uint16_t RegSize,
uint8_t * pBuffer,
uint16_t Length
)

Reads multiple data on the BUS.

Parameters:

Addr : I2C Address

Reg : Reg Address

RegSize : The target register size (can be 8BIT or 16BIT)

pBuffer : pointer to read data buffer

Length : length of the data

Return values:

0 if no problems to read multiple data

Definition at line **297** of file **stm32l0xx_nucleo_32.c**.

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

void I2C1_Write (uint8_t Addr,
uint8_t Reg,
uint8_t Value
)

Writes a single data.

Parameters:

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

Return values:

None

Definition at line **250** of file **stm32l0xx_nucleo_32.c**.

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

HAL_StatusTypeDef I2C1_WriteBuffer (uint16_t Addr,
uint8_t Reg,
uint16_t RegSize,
uint8_t * pBuffer,
uint16_t Length
)

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

Parameters:

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

Return values:

None

Definition at line **333** of file **stm32l0xx_nucleo_32.c**.

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

STM32L0xx_Nucleo_32 BSP User Manual

Exported Types

NUCLEO 32

Enumerations

enum	**Led_TypeDef** { **LED3** = 0, **LED_GREEN** = LED3 }

enum	**JOYState_TypeDef** {
	JOY_NONE = 0, **JOY_SEL** = 1, **JOY_DOWN** = 2,
	JOY_LEFT = 3,
	JOY_RIGHT = 4, **JOY_UP** = 5
	}

Enumeration Type Documentation

enum JOYState_TypeDef

Enumerator:

JOY_NONE
JOY_SEL
JOY_DOWN
JOY_LEFT
JOY_RIGHT
JOY_UP

Definition at line **69** of file **stm32l0xx_nucleo_32.h**.

enum Led_TypeDef

Enumerator:

LED3
LED_GREEN

Definition at line **63** of file **stm32l0xx_nucleo_32.h**.

STM32L0xx_Nucleo_32 BSP User Manual

STM32L0XX_NUCLEO_32_COMPONENT

Exported Constants

Defines

#define	**SD_CS_LOW**() HAL_GPIO_WritePin(**SD_CS_GPIO_PORT**, GPIO_PIN_RESET) SD Control Lines management.
#define	**SD_CS_HIGH**() HAL_GPIO_WritePin(**SD_CS_GPIO_PORT**, GPIO_PIN_SET)
#define	**LCD_CS_LOW**() HAL_GPIO_WritePin(**LCD_CS_GPIO_POR** GPIO_PIN_RESET) LCD Control Lines management.
#define	**LCD_CS_HIGH**() HAL_GPIO_WritePin(**LCD_CS_GPIO_POR** GPIO_PIN_SET)
#define	**LCD_DC_LOW**() HAL_GPIO_WritePin(**LCD_DC_GPIO_POR** GPIO_PIN_RESET)
#define	**LCD_DC_HIGH**() HAL_GPIO_WritePin(**LCD_DC_GPIO_POR** GPIO_PIN_SET)
#define	**SD_CS_PIN** GPIO_PIN_5 SD Control Interface pins.
#define	**SD_CS_GPIO_PORT** GPIOB
#define	**SD_CS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_CLK_
#define	**SD_CS_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_CLK_
#define	**LCD_CS_PIN** GPIO_PIN_6 LCD Control Interface pins.
#define	**LCD_CS_GPIO_PORT** GPIOB
#define	**LCD_CS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_CLK
#define	**LCD_CS_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_CL
#define	**LCD_DC_PIN** GPIO_PIN_9 LCD Data/Command Interface pins.
#define	**LCD_DC_GPIO_PORT** GPIOA
#define	**LCD_DC_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOA_CLK
#define	**LCD_DC_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOA_CL
#define	**NUCLEO_ADCx** ADC1 ADC Interface pins used to detect motion of Joystick available

TFT shield.

```
#define NUCLEO_ADCx_CLK_ENABLE()   __HAL_RCC_ADC1_CLK
#define NUCLEO_ADCx_GPIO_PORT   GPIOB
#define NUCLEO_ADCx_GPIO_PIN   GPIO_PIN_0
#define NUCLEO_ADCx_GPIO_CLK_ENABLE()   __HAL_RCC_GPIC
#define NUCLEO_ADCx_GPIO_CLK_DISABLE()   __HAL_RCC_GPI
```

Define Documentation

#define LCD_CS_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOB_C

Definition at line 206 of file stm32l0xx_nucleo_32.h.

#define LCD_CS_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOB_CL

Definition at line 205 of file stm32l0xx_nucleo_32.h.

Referenced by LCD_IO_Init().

#define LCD_CS_GPIO_PORT GPIOB

Definition at line 204 of file stm32l0xx_nucleo_32.h.

Referenced by LCD_IO_Init().

#define LCD_CS_HIGH ( ) HAL_GPIO_WritePin(LCD_CS_GPIO_P

Definition at line 188 of file stm32l0xx_nucleo_32.h.

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

#define LCD_CS_LOW ( ) HAL_GPIO_WritePin(LCD_CS_GPIO_P

LCD Control Lines management.

Definition at line 187 of file stm32l0xx_nucleo_32.h.

Referenced by LCD_IO_WriteData(), LCD_IO_WriteMultipleData(),

and **LCD_IO_WriteReg()**.

#define LCD_CS_PIN GPIO_PIN_6

LCD Control Interface pins.

Definition at line **203** of file **stm32l0xx_nucleo_32.h**.

Referenced by **LCD_IO_Init()**.

#define LCD_DC_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOA_C

Definition at line **214** of file **stm32l0xx_nucleo_32.h**.

#define LCD_DC_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOA_CL

Definition at line **213** of file **stm32l0xx_nucleo_32.h**.

Referenced by **LCD_IO_Init()**.

#define LCD_DC_GPIO_PORT GPIOA

Definition at line **212** of file **stm32l0xx_nucleo_32.h**.

Referenced by **LCD_IO_Init()**.

#define LCD_DC_HIGH ( ) HAL_GPIO_WritePin(LCD_DC_GPIO_P

Definition at line **190** of file **stm32l0xx_nucleo_32.h**.

Referenced by **LCD_IO_WriteData()**, and **LCD_IO_WriteMultipleData()**.

#define LCD_DC_LOW ( ) HAL_GPIO_WritePin(LCD_DC_GPIO_P(

Definition at line 189 of file stm32l0xx_nucleo_32.h.

Referenced by LCD_IO_WriteReg().

#define LCD_DC_PIN GPIO_PIN_9

LCD Data/Command Interface pins.

Definition at line 211 of file stm32l0xx_nucleo_32.h.

Referenced by LCD_IO_Init().

#define NUCLEO_ADCx ADC1

ADC Interface pins used to detect motion of Joystick available on Adafruit 1.8" TFT shield.

Definition at line 222 of file stm32l0xx_nucleo_32.h.

Referenced by ADCx_Init().

#define NUCLEO_ADCx_CLK_ENABLE ( ) __HAL_RCC_ADC1_CL

Definition at line 223 of file stm32l0xx_nucleo_32.h.

Referenced by ADCx_MspInit().

#define NUCLEO_ADCx_GPIO_CLK_DISABLE ( ) __HAL_RCC_GI

Definition at line 228 of file stm32l0xx_nucleo_32.h.

#define NUCLEO_ADCx_GPIO_CLK_ENABLE ( ) __HAL_RCC_GP

Definition at line 227 of file stm32l0xx_nucleo_32.h.

Referenced by ADCx_MspInit().

#define NUCLEO_ADCx_GPIO_PIN GPIO_PIN_0

Definition at line 226 of file stm32l0xx_nucleo_32.h.

Referenced by ADCx_MspInit().

#define NUCLEO_ADCx_GPIO_PORT GPIOB

Definition at line 225 of file stm32l0xx_nucleo_32.h.

Referenced by ADCx_MspInit().

#define SD_CS_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOB_CL

Definition at line 198 of file stm32l0xx_nucleo_32.h.

#define SD_CS_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOB_CLK

Definition at line 197 of file stm32l0xx_nucleo_32.h.

Referenced by SD_IO_Init().

#define SD_CS_GPIO_PORT GPIOB

Definition at line 196 of file stm32l0xx_nucleo_32.h.

Referenced by **SD_IO_Init()**.

#define SD_CS_HIGH ( ) HAL_GPIO_WritePin(SD_CS_GPIO_POR

Definition at line **182** of file **stm32l0xx_nucleo_32.h**.

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

#define SD_CS_LOW ( ) HAL_GPIO_WritePin(SD_CS_GPIO_POR

SD Control Lines management.

Definition at line **181** of file **stm32l0xx_nucleo_32.h**.

Referenced by **SD_IO_WriteCmd()**.

#define SD_CS_PIN GPIO_PIN_5

SD Control Interface pins.

Definition at line **195** of file **stm32l0xx_nucleo_32.h**.

Referenced by **SD_IO_Init()**.

STM32L0xx_Nucleo_32 BSP User Manual

STM32L0XX_NUCLEO_32_LED

Exported Constants

Define for STM32L0XX_NUCLEO_32 board. More...

Defines

#define	**LEDn** 1
#define	**LED3_PIN** GPIO_PIN_3
#define	**LED3_GPIO_PORT** GPIOB
#define	**LED3_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_CLK_E
#define	**LED3_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_CLK_D
#define	**LEDx_GPIO_CLK_ENABLE**(__INDEX__) do {**LED3_GPIO_CLK_ENABLE**(); } while(0)
#define	**LEDx_GPIO_CLK_DISABLE**(__INDEX__) **LED3_GPIO_CLK**

Detailed Description

Define for STM32L0XX_NUCLEO_32 board.

Define Documentation

#define LED3_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOB_CLK

Definition at line 102 of file stm32l0xx_nucleo_32.h.

#define LED3_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOB_CLK_

Definition at line 101 of file stm32l0xx_nucleo_32.h.

#define LED3_GPIO_PORT GPIOB

Definition at line 100 of file stm32l0xx_nucleo_32.h.

#define LED3_PIN GPIO_PIN_3

Definition at line 99 of file stm32l0xx_nucleo_32.h.

#define LEDn 1

Definition at line 97 of file stm32l0xx_nucleo_32.h.

#define LEDx_GPIO_CLK_DISABLE (__INDEX__) LED3_GPIO_C

Definition at line 105 of file stm32l0xx_nucleo_32.h.

#define LEDx_GPIO_CLK_ENABLE (__INDEX__) do {LED3_GPI

Definition at line 104 of file stm32l0xx_nucleo_32.h.

Referenced by **BSP_LED_Init()**.

STM32L0xx_Nucleo_32 BSP User Manual

Firmware Directory Reference

Directories

directory	**Drivers**

STM32L0xx_Nucleo_32 BSP User Manual

Drivers Directory Reference

Directories

directory	**BSP**

STM32L0xx_Nucleo_32 BSP User Manual

BSP Directory Reference

Directories

directory	**STM32L0xx_Nucleo_32**

STM32L0xx_Nucleo_32 BSP User Manual

STM32L0xx_Nucleo_32 Directory Reference

Files

file	**stm32l0xx_nucleo_32.c** [code] This file provides set of firmware functions to manage:
file	**stm32l0xx_nucleo_32.h** [code] This file contains definitions for:

STM32L0xx_Nucleo_32 BSP User Manual

stm32l0xx_nucleo_32.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l0xx_nucleo_32.h
00004   * @author  MCD Application Team
00005   * @brief   This file contains definitions for:
00006   *          - LEDs and push-button available on STM32L0XX-Nucleo Kit
00007   *            from STMicroelectronics
00008   *          - LCD, joystick and microSD available on Adafruit 1.8" TFT LCD
00009   *            shield (reference ID 802)
00010   ******************************************************************************
00011   * @attention
00012   *
00013   * <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>
00014   *
00015   * Redistribution and use in source and binary forms, with or without modification,
00016   * are permitted provided that the following conditions are met:
```

```
00017   *   1. Redistributions of source code must
 retain the above copyright notice,
00018   *      this list of conditions and the fol
lowing disclaimer.
00019   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00020   *      this list of conditions and the fol
lowing disclaimer in the documentation
00021   *      and/or other materials provided wit
h the distribution.
00022   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00023   *      may be used to endorse or promote p
roducts derived from this software
00024   *      without specific prior written perm
ission.
00025   *
00026   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00027   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00028   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00029   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00030   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00031   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
00032   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00033   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00034   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00035   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
```

```
00036   *
00037   ******************************************
*********************************
00038   */
00039
00040 /* Define to prevent recursive inclusion ---
-----------------------------------*/
00041 #ifndef __STM32L0XX_NUCLEO_32_H
00042 #define __STM32L0XX_NUCLEO_32_H
00043
00044 #ifdef __cplusplus
00045  extern "C" {
00046 #endif
00047
00048 /* Includes ------------------------------
-----------------------------------*/
00049 #include "stm32l0xx_hal.h"
00050
00051
00052 /** @addtogroup BSP
00053   * @{
00054   */
00055
00056 /** @addtogroup STM32L0XX_NUCLEO_32 NUCLEO 32

00057   * @{
00058   */
00059
00060 /** @defgroup STM32L0XX_NUCLEO_32_Exported_T
ypes Exported Types
00061   * @{
00062   */
00063 typedef enum
00064 {
00065   LED3 = 0,
00066   LED_GREEN = LED3
00067 } Led_TypeDef;
```

```
00068
00069 typedef enum
00070 {
00071   JOY_NONE  = 0,
00072   JOY_SEL   = 1,
00073   JOY_DOWN  = 2,
00074   JOY_LEFT  = 3,
00075   JOY_RIGHT = 4,
00076   JOY_UP    = 5
00077 } JOYState_TypeDef;
00078
00079 /**
00080   * @}
00081   */
00082
00083 /** @defgroup STM32L0XX_NUCLEO_32_Exported_C
onstants Exported Constants
00084   * @{
00085   */
00086
00087 /**
00088 * @brief        Define for STM32L0XX_NUCLEO_
32 board
00089   */
00090 #if !defined (USE_STM32L0XX_NUCLEO_32)
00091  #define USE_STM32L0XX_NUCLEO_32
00092 #endif
00093
00094 /** @addtogroup STM32L0XX_NUCLEO_32_LED
00095   * @{
00096   */
00097 #define LEDn                                    1
00098
00099 #define LED3_PIN                                G
PIO_PIN_3
00100 #define LED3_GPIO_PORT                          G
PIOB
```

```
00101 #define LED3_GPIO_CLK_ENABLE()             __H
AL_RCC_GPIOB_CLK_ENABLE()
00102 #define LED3_GPIO_CLK_DISABLE()            __H
AL_RCC_GPIOB_CLK_DISABLE()
00103
00104 #define LEDx_GPIO_CLK_ENABLE(__INDEX__)    d
o {LED3_GPIO_CLK_ENABLE(); } while(0)
00105 #define LEDx_GPIO_CLK_DISABLE(__INDEX__)   L
ED3_GPIO_CLK_DISABLE())
00106 /**
00107   * @}
00108   */
00109
00110
00111
00112 /** @addtogroup STM32L0XX_NUCLEO_32_BUS
00113   * @{
00114   */
00115 #if defined(HAL_I2C_MODULE_ENABLED)
00116 /*##################### I2C2 ###############
####################*/
00117 /* User can use this section to tailor I2Cx
instance used and associated resources */
00118 /* Definition for I2C1 Pins */
00119 #define BSP_I2C1                           I2C1
00120 #define BSP_I2C1_CLK_ENABLE()              __HA
L_RCC_I2C1_CLK_ENABLE()
00121 #define BSP_I2C1_CLK_DISABLE()             __HA
L_RCC_I2C1_CLK_DISABLE()
00122 #define BSP_I2C1_FORCE_RESET()             __HA
L_RCC_I2C1_FORCE_RESET()
00123 #define BSP_I2C1_RELEASE_RESET()           __HA
L_RCC_I2C1_RELEASE_RESET()
00124
00125 #define BSP_I2C1_SCL_PIN                   GPIO
_PIN_6    /* PB.6 add wire between D5 and A5 */
00126 #define BSP_I2C1_SDA_PIN                   GPIO
```

```
_PIN_7    /* PB.7 add wire between D4 and A4 */
00127
00128 #define BSP_I2C1_GPIO_PORT              GPIO
B      /* GPIOB */
00129 #define BSP_I2C1_GPIO_CLK_ENABLE()      __HA
L_RCC_GPIOB_CLK_ENABLE()
00130 #define BSP_I2C1_GPIO_CLK_DISABLE()     __HA
L_RCC_GPIOB_CLK_DISABLE()
00131 #define BSP_I2C1_SCL_SDA_AF             GPIO
_AF1_I2C1
00132
00133 /* Maximum Timeout values for flags waiting
loops. These timeouts are not based
00134    on accurate values, they just guarantee t
hat the application will not remain
00135    stuck if the I2C communication is corrupt
ed.
00136    You may modify these timeout values depen
ding on CPU frequency and application
00137    conditions (interrupts routines ...). */

00138 #define BSP_I2C1_TIMEOUT_MAX            1000
00139
00140 /* I2C TIMING is calculated in case of the I
2C Clock source is the SYSCLK = 32 MHz */
00141 /* Set TIMING to 0x009080B5 to reach 100 KHz
 speed (Rise time = 50ns, Fall time = 10ns) */
00142 #define I2C1_TIMING                      0x0
09080B5
00143
00144 #endif /* HAL_I2C_MODULE_ENABLED */
00145
00146 #if defined(HAL_SPI_MODULE_ENABLED)
00147 /*########################## SPI1 ############
######################*/
00148 #define NUCLEO_SPIx
        SPI1
```

```
00149 #define NUCLEO_SPIx_CLK_ENABLE()
      __HAL_RCC_SPI1_CLK_ENABLE()
00150
00151 #define NUCLEO_SPIx_SCK_AF
        GPIO_AF0_SPI1
00152 #define NUCLEO_SPIx_SCK_GPIO_PORT
        GPIOA
00153 #define NUCLEO_SPIx_SCK_PIN
        GPIO_PIN_11
00154 #define NUCLEO_SPIx_SCK_GPIO_CLK_ENABLE()
      __HAL_RCC_GPIOA_CLK_ENABLE()
00155 #define NUCLEO_SPIx_SCK_GPIO_CLK_DISABLE()
      __HAL_RCC_GPIOA_CLK_DISABLE()
00156
00157 #define NUCLEO_SPIx_MISO_MOSI_AF
        GPIO_AF0_SPI1
00158 #define NUCLEO_SPIx_MISO_MOSI_GPIO_PORT
        GPIOB
00159 #define NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_ENABL
E()   __HAL_RCC_GPIOB_CLK_ENABLE()
00160 #define NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_DISAB
LE()  __HAL_RCC_GPIOB_CLK_DISABLE()
00161 #define NUCLEO_SPIx_MISO_PIN
        GPIO_PIN_4
00162 #define NUCLEO_SPIx_MOSI_PIN
        GPIO_PIN_5
00163 /* Maximum Timeout values for flags waiting
loops. These timeouts are not based
00164    on accurate values, they just guarantee t
hat the application will not remain
00165    stuck if the SPI communication is corrupt
ed.
00166    You may modify these timeout values depen
ding on CPU frequency and application
00167    conditions (interrupts routines ...). */

00168 #define NUCLEO_SPIx_TIMEOUT_MAX
```

```
      1000
00169 #endif /* HAL_SPI_MODULE_ENABLED */
00170 /**
00171   * @}
00172   */
00173
00174 /** @addtogroup STM32L0XX_NUCLEO_32_COMPONENT

00175   * @{
00176   */
00177
00178 /**
00179   * @brief  SD Control Lines management
00180   */
00181 #define SD_CS_LOW()       HAL_GPIO_WritePin(
SD_CS_GPIO_PORT, SD_CS_PIN, GPIO_PIN_RESET)
00182 #define SD_CS_HIGH()      HAL_GPIO_WritePin(
SD_CS_GPIO_PORT, SD_CS_PIN, GPIO_PIN_SET)
00183
00184 /**
00185   * @brief  LCD Control Lines management
00186   */
00187 #define LCD_CS_LOW()      HAL_GPIO_WritePin(
LCD_CS_GPIO_PORT, LCD_CS_PIN, GPIO_PIN_RESET)
00188 #define LCD_CS_HIGH()     HAL_GPIO_WritePin(
LCD_CS_GPIO_PORT, LCD_CS_PIN, GPIO_PIN_SET)
00189 #define LCD_DC_LOW()      HAL_GPIO_WritePin(
LCD_DC_GPIO_PORT, LCD_DC_PIN, GPIO_PIN_RESET)
00190 #define LCD_DC_HIGH()     HAL_GPIO_WritePin(
LCD_DC_GPIO_PORT, LCD_DC_PIN, GPIO_PIN_SET)
00191
00192 /**
00193   * @brief  SD Control Interface pins
00194   */
00195 #define SD_CS_PIN
      GPIO_PIN_5
00196 #define SD_CS_GPIO_PORT
```

```
      GPIOB
00197 #define SD_CS_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOB_CLK_ENABLE()
00198 #define SD_CS_GPIO_CLK_DISABLE()
    __HAL_RCC_GPIOB_CLK_DISABLE()
00199
00200 /**
00201   * @brief  LCD Control Interface pins
00202   */
00203 #define LCD_CS_PIN
       GPIO_PIN_6
00204 #define LCD_CS_GPIO_PORT
       GPIOB
00205 #define LCD_CS_GPIO_CLK_ENABLE()
     __HAL_RCC_GPIOB_CLK_ENABLE()
00206 #define LCD_CS_GPIO_CLK_DISABLE()
     __HAL_RCC_GPIOB_CLK_DISABLE()
00207
00208 /**
00209   * @brief  LCD Data/Command Interface pins
00210   */
00211 #define LCD_DC_PIN
       GPIO_PIN_9
00212 #define LCD_DC_GPIO_PORT
       GPIOA
00213 #define LCD_DC_GPIO_CLK_ENABLE()
     __HAL_RCC_GPIOA_CLK_ENABLE()
00214 #define LCD_DC_GPIO_CLK_DISABLE()
     __HAL_RCC_GPIOA_CLK_DISABLE()
00215
00216 #if defined(HAL_ADC_MODULE_ENABLED)
00217 /*###################### ADC1 ###############
######################*/
00218 /**
00219   * @brief  ADC Interface pins
00220   *         used to detect motion of Joystic
k available on Adafruit 1.8" TFT shield
```

```
00221   */
00222 #define NUCLEO_ADCx
        ADC1
00223 #define NUCLEO_ADCx_CLK_ENABLE()
      __HAL_RCC_ADC1_CLK_ENABLE()
00224
00225 #define NUCLEO_ADCx_GPIO_PORT
        GPIOB
00226 #define NUCLEO_ADCx_GPIO_PIN
        GPIO_PIN_0
00227 #define NUCLEO_ADCx_GPIO_CLK_ENABLE()
      __HAL_RCC_GPIOB_CLK_ENABLE()
00228 #define NUCLEO_ADCx_GPIO_CLK_DISABLE()
      __HAL_RCC_GPIOB_CLK_ENABLE()
00229
00230 #endif /* HAL_ADC_MODULE_ENABLED */
00231 /**
00232   * @}
00233   */
00234
00235 /** @defgroup STM32L0XX_NUCLEO_32_Exported_M
acros Exported Macros
00236   * @{
00237   */
00238 /**
00239   * @}
00240   */
00241
00242 /** @defgroup STM32L0XX_NUCLEO_32_Internal_F
unctions Internal Functions
00243   * @{
00244   */
00245 #if defined(HAL_I2C_MODULE_ENABLED)
00246 /* I2C1 bus function */
00247 /* Link function for I2C peripherals */
00248 void              I2C1_Init(void);
00249 void              I2C1_Error (void);
```

```
00250 void                I2C1_MspInit(I2C_HandleTy
peDef *hi2c);
00251 void                I2C1_Write(uint8_t Addr,
uint8_t Reg, uint8_t Value);
00252 uint8_t             I2C1_Read(uint8_t Addr, u
int8_t Reg);
00253 HAL_StatusTypeDef  I2C1_WriteBuffer(uint16_t
 Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBu
ffer, uint16_t Length);
00254 HAL_StatusTypeDef  I2C1_ReadBuffer(uint16_t
Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuf
fer, uint16_t Length);
00255 HAL_StatusTypeDef  I2C1_IsDeviceReady(uint16
_t DevAddress, uint32_t Trials);
00256 #endif /* HAL_I2C_MODULE_ENABLED */
00257 /**
00258   * @}
00259   */
00260
00261 /** @defgroup STM32L0XX_NUCLEO_32_Exported_F
unctions Exported Functions
00262   * @{
00263   */
00264 uint32_t         BSP_GetVersion(void);
00265 void             BSP_LED_Init(Led_TypeDef Le
d);
00266 void             BSP_LED_On(Led_TypeDef Led)
;
00267 void             BSP_LED_Off(Led_TypeDef Led
);
00268 void             BSP_LED_Toggle(Led_TypeDef
Led);
00269 #if defined(HAL_ADC_MODULE_ENABLED)
00270 uint8_t          BSP_JOY_Init(void);
00271 JOYState_TypeDef BSP_JOY_GetState(void);
00272 #endif /* HAL_ADC_MODULE_ENABLED */
00273 /**
```

```
00274   * @}
00275   */
00276
00277 /**
00278   * @}
00279   */
00280
00281 /**
00282   * @}
00283   */
00284
00285 /**
00286   * @}
00287   */
00288
00289 #ifdef __cplusplus
00290 }
00291 #endif
00292
00293 #endif /* __STM32L0XX_NUCLEO_32_H */
00294
00295 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
00296
```

STM32L0xx_Nucleo_32 BSP User Manual

stm32l0xx_nucleo_32.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************
******************************
00003   * @file    stm32l0xx_nucleo_32.c
00004   * @author  MCD Application Team
00005   * @brief   This file provides set of firmw
are functions to manage:
00006   *          - LEDs and push-button availabl
e on STM32L0XX-Nucleo Kit
00007   *            from STMicroelectronics
00008   ******************************************
******************************
00009   * @attention
00010   *
00011   * <h2><center>&copy; COPYRIGHT(c) 2016 STM
icroelectronics</center></h2>
00012   *
00013   * Redistribution and use in source and bin
ary forms, with or without modification,
00014   * are permitted provided that the followin
g conditions are met:
00015   *   1. Redistributions of source code must
 retain the above copyright notice,
00016   *      this list of conditions and the fol
```

```
lowing disclaimer.
00017   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00018   *      this list of conditions and the fol
lowing disclaimer in the documentation
00019   *      and/or other materials provided wit
h the distribution.
00020   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00021   *      may be used to endorse or promote p
roducts derived from this software
00022   *      without specific prior written perm
ission.
00023   *
00024   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00025   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00026   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00027   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00028   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00029   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
00030   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00031   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00032   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00033   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00034   *
00035   ****************************************
**********************************
```

```
00036   */
00037
00038 /* Includes ------------------------------
------------------------------------*/
00039 #include "stm32l0xx_nucleo_32.h"
00040
00041 /** @addtogroup BSP
00042   * @{
00043   */
00044
00045 /** @addtogroup STM32L0XX_NUCLEO_32 NUCLEO 32

00046   * @brief This file provides set of firmwar
e functions to manage Leds and push-button
00047   *        available on STM32L0XX-Nucleo Kit
 from STMicroelectronics.
00048   * @{
00049   */
00050
00051 /** @defgroup STM32L0XX_NUCLEO_32_Private_De
fines Private Defines
00052   * @{
00053   */
00054
00055 /**
00056   * @brief STM32L0XX NUCLEO BSP Driver versi
on number
00057   */
00058 #define __STM32L0XX_NUCLEO_32_BSP_VERSION_MA
IN   (0x01) /*!< [31:24] main version */
00059 #define __STM32L0XX_NUCLEO_32_BSP_VERSION_SU
B1   (0x00) /*!< [23:16] sub1 version */
00060 #define __STM32L0XX_NUCLEO_32_BSP_VERSION_SU
B2   (0x03) /*!< [15:8]  sub2 version */
00061 #define __STM32L0XX_NUCLEO_32_BSP_VERSION_RC
     (0x00) /*!< [7:0]  release candidate */
00062 #define __STM32L0XX_NUCLEO_32_BSP_VERSION
```

```
      ((__STM32L0XX_NUCLEO_32_BSP_VERSION_MAIN <<
24)\
00063
 |(__STM32L0XX_NUCLEO_32_BSP_VERSION_SUB1 << 16)\
00064
 |(__STM32L0XX_NUCLEO_32_BSP_VERSION_SUB2 << 8 )\
00065
 |(__STM32L0XX_NUCLEO_32_BSP_VERSION_RC))
00066
00067 /**
00068   * @brief LINK SD Card
00069   */
00070 #define SD_DUMMY_BYTE            0xFF
00071 #define SD_NO_RESPONSE_EXPECTED  0x80
00072
00073 /**
00074   * @}
00075   */
00076
00077 /** @defgroup STM32L0XX_NUCLEO_32_Private_Va
riables Private Variables
00078   * @{
00079   */
00080 GPIO_TypeDef* LED_PORT[LEDn] = {LED3_GPIO_PO
RT};
00081 const uint16_t LED_PIN[LEDn] = {LED3_PIN};
00082
00083 /**
00084  * @brief BUS variables
00085  */
00086 #if defined(HAL_I2C_MODULE_ENABLED)
00087 uint32_t I2c1Timeout = BSP_I2C1_TIMEOUT_MAX;
    /*<! Value of Timeout when I2C1 communication
fails */
00088 I2C_HandleTypeDef heval_I2c1;
00089 #endif /* HAL_I2C_MODULE_ENABLED */
00090
```

```
00091 #ifdef HAL_SPI_MODULE_ENABLED
00092 uint32_t SpixTimeout = NUCLEO_SPIx_TIMEOUT_M
AX; /*<! Value of Timeout when SPI communication f
ails */
00093 static SPI_HandleTypeDef hnucleo_Spi;
00094 #endif /* HAL_SPI_MODULE_ENABLED */
00095
00096 #ifdef HAL_ADC_MODULE_ENABLED
00097 static ADC_HandleTypeDef hnucleo_Adc;
00098 /* ADC channel configuration structure decla
ration */
00099 static ADC_ChannelConfTypeDef sConfig;
00100 #endif /* HAL_ADC_MODULE_ENABLED */
00101 /**
00102   * @}
00103   */
00104
00105 /** @defgroup STM32L0XX_NUCLEO_32_Private_Fu
nction_Prototypes Private Function Prototypes
00106   * @{
00107   */
00108
00109 #ifdef HAL_SPI_MODULE_ENABLED
00110 static void                SPIx_Init(void);
00111 static void                SPIx_Write(uint8_t
 Value);
00112 static uint32_t            SPIx_Read(void);
00113 static void                SPIx_Error (void);
00114 static void                SPIx_MspInit(SPI_H
andleTypeDef *hspi);
00115
00116 /* SD IO functions */
00117 void                       SD_IO_Init(void);
00118 HAL_StatusTypeDef          SD_IO_WriteCmd(uin
t8_t Cmd, uint32_t Arg, uint8_t Crc, uint8_t Respo
nse);
00119 HAL_StatusTypeDef          SD_IO_WaitResponse
```

```
(uint8_t Response);
00120 void                    SD_IO_WriteDummy(v
oid);
00121 void                    SD_IO_WriteByte(ui
nt8_t Data);
00122 uint8_t                 SD_IO_ReadByte(void
);
00123
00124 /* LCD IO functions */
00125 void                    LCD_IO_Init(void);
00126 void                    LCD_IO_WriteData(u
int8_t Data);
00127 void                    LCD_IO_WriteMultip
leData(uint8_t *pData, uint32_t Size);
00128 void                    LCD_IO_WriteReg(ui
nt8_t LCDReg);
00129 void                    LCD_Delay(uint32_t
 delay);
00130 #endif /* HAL_SPI_MODULE_ENABLED */
00131
00132 #ifdef HAL_ADC_MODULE_ENABLED
00133 static void             ADCx_Init(void);
00134 static void             ADCx_MspInit(ADC_H
andleTypeDef *hadc);
00135 #endif /* HAL_ADC_MODULE_ENABLED */
00136 /**
00137   * @}
00138   */
00139
00140 /** @defgroup STM32L0XX_NUCLEO_32_Private_Fu
nctions Private Functions
00141   * @{
00142   */
00143
00144 /**
00145   * @brief  This method returns the STM32L0X
X NUCLEO BSP Driver revision
```

```
00146   * @retval version : 0xXYZR (8bits for each
 decimal, R for RC)
00147   */
00148 uint32_t BSP_GetVersion(void)
00149 {
00150   return __STM32L0XX_NUCLEO_32_BSP_VERSION;
00151 }
00152
00153 /**
00154   * @brief  Configures LED GPIO.
00155   * @param  Led: Specifies the Led to be con
figured.
00156   *   This parameter can be one of following
 parameters:
00157   *            @arg  LED3
00158   * @retval None
00159   */
00160 void BSP_LED_Init(Led_TypeDef Led)
00161 {
00162   GPIO_InitTypeDef  GPIO_InitStruct;
00163
00164   /* Enable the GPIO_LED Clock */
00165   LEDx_GPIO_CLK_ENABLE(Led);
00166
00167   /* Configure the GPIO_LED pin */
00168   GPIO_InitStruct.Pin = LED_PIN[Led];
00169   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP
;
00170   GPIO_InitStruct.Pull = GPIO_NOPULL;
00171   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VE
RY_HIGH;
00172
00173   HAL_GPIO_Init(LED_PORT[Led], &GPIO_InitStr
uct);
00174   HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[L
ed], GPIO_PIN_RESET);
00175 }
```

```
00176
00177 /**
00178   * @brief  Turns selected LED On.
00179   * @param  Led: Specifies the Led to be set
 on.
00180   *   This parameter can be one of following
 parameters:
00181   *            @arg  LED3
00182   * @retval None
00183   */
00184 void BSP_LED_On(Led_TypeDef Led)
00185 {
00186   HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[L
ed], GPIO_PIN_SET);
00187 }
00188
00189 /**
00190   * @brief  Turns selected LED Off.
00191   * @param  Led: Specifies the Led to be set
 off.
00192   *   This parameter can be one of following
 parameters:
00193   *            @arg  LED3
00194   * @retval None
00195   */
00196 void BSP_LED_Off(Led_TypeDef Led)
00197 {
00198   HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[L
ed], GPIO_PIN_RESET);
00199 }
00200
00201 /**
00202   * @brief  Toggles the selected LED.
00203   * @param  Led: Specifies the Led to be tog
gled.
00204   *   This parameter can be one of following
 parameters:
```

```
00205   *            @arg  LED3
00206   * @retval None
00207   */
00208 void BSP_LED_Toggle(Led_TypeDef Led)
00209 {
00210   HAL_GPIO_TogglePin(LED_PORT[Led], LED_PIN[
Led]);
00211 }
00212
00213 /**********************************************
**********************************
00214                                   BUS OPERATIONS
00215 **********************************************
**********************************/
00216 #if defined(HAL_I2C_MODULE_ENABLED)
00217 /****************************** I2C Routine
s ********************************/
00218
00219 /**
00220   * @brief I2C Bus initialization
00221   * @retval None
00222   */
00223 void I2C1_Init(void)
00224 {
00225   if(HAL_I2C_GetState(&heval_I2c1) == HAL_I2
C_STATE_RESET)
00226   {
00227     heval_I2c1.Instance              = BSP_I
2C1;
00228     heval_I2c1.Init.Timing           = I2C1_
TIMING;
00229     heval_I2c1.Init.OwnAddress1      = 0;
00230     heval_I2c1.Init.AddressingMode   = I2C_A
DDRESSINGMODE_7BIT;
00231     heval_I2c1.Init.DualAddressMode  = I2C_D
UALADDRESS_DISABLE;
00232     heval_I2c1.Init.OwnAddress2      = 0;
```

```
00233     heval_I2c1.Init.OwnAddress2Masks = I2C_O
A2_NOMASK;
00234     heval_I2c1.Init.GeneralCallMode  = I2C_G
ENERALCALL_DISABLE;
00235     heval_I2c1.Init.NoStretchMode    = I2C_N
OSTRETCH_DISABLE;
00236
00237     /* Init the I2C */
00238     I2C1_MspInit(&heval_I2c1);
00239     HAL_I2C_Init(&heval_I2c1);
00240   }
00241 }
00242
00243 /**
00244   * @brief  Writes a single data.
00245   * @param  Addr: I2C address
00246   * @param  Reg: Register address
00247   * @param  Value: Data to be written
00248   * @retval None
00249   */
00250 void I2C1_Write(uint8_t Addr, uint8_t Reg, u
int8_t Value)
00251 {
00252   HAL_StatusTypeDef status = HAL_OK;
00253
00254   status = HAL_I2C_Mem_Write(&heval_I2c1, Ad
dr, (uint16_t)Reg, I2C_MEMADD_SIZE_8BIT, &Value, 1
, 100);
00255
00256   /* Check the communication status */
00257   if(status != HAL_OK)
00258   {
00259     /* Execute user timeout callback */
00260     I2C1_Error();
00261   }
00262 }
00263
```

```
00264 /**
00265   * @brief  Reads a single data.
00266   * @param  Addr: I2C address
00267   * @param  Reg: Register address
00268   * @retval Read data
00269   */
00270 uint8_t I2C1_Read(uint8_t Addr, uint8_t Reg)
00271 {
00272   HAL_StatusTypeDef status = HAL_OK;
00273   uint8_t Value = 0;
00274
00275   status = HAL_I2C_Mem_Read(&heval_I2c1, Add
r, Reg, I2C_MEMADD_SIZE_8BIT, &Value, 1, 1000);
00276
00277   /* Check the communication status */
00278   if(status != HAL_OK)
00279   {
00280     /* Execute user timeout callback */
00281     I2C1_Error();
00282   }
00283   return Value;
00284 }
00285
00286
00287
00288 /**
00289   * @brief  Reads multiple data on the BUS.
00290   * @param  Addr  : I2C Address
00291   * @param  Reg   : Reg Address
00292   * @param  RegSize : The target register si
ze (can be 8BIT or 16BIT)
00293   * @param  pBuffer : pointer to read data b
uffer
00294   * @param  Length : length of the data
00295   * @retval 0 if no problems to read multipl
e data
00296   */
```

```
00297 HAL_StatusTypeDef I2C1_ReadBuffer(uint16_t A
ddr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuff
er, uint16_t Length)
00298 {
00299   HAL_StatusTypeDef status = HAL_OK;
00300
00301   status = HAL_I2C_Mem_Read(&heval_I2c1, Add
r, Reg, RegSize, pBuffer, Length, I2c1Timeout);
00302
00303   /* Check the communication status */
00304   if(status != HAL_OK)
00305   {
00306     /* Re-Initiaize the BUS */
00307     I2C1_Error();
00308   }
00309   return status;
00310 }
00311
00312 /**
00313   * @brief  Checks if target device is ready
 for communication.
00314   * @note   This function is used with Memor
y devices
00315   * @param  DevAddress: Target device address

00316   * @param  Trials: Number of trials
00317   * @retval HAL status
00318   */
00319 HAL_StatusTypeDef I2C1_IsDeviceReady(uint16_
t DevAddress, uint32_t Trials)
00320 {
00321   return (HAL_I2C_IsDeviceReady(&heval_I2c1,
 DevAddress, Trials, I2c1Timeout));
00322 }
00323
00324 /**
00325   * @brief  Write a value in a register of t
```

```
he device through BUS.
00326   * @param  Addr: Device address on BUS Bus.

00327   * @param  Reg: The target register address
 to write
00328   * @param  RegSize: The target register siz
e (can be 8BIT or 16BIT)
00329   * @param  pBuffer: The target register val
ue to be written
00330   * @param  Length: buffer size to be written

00331   * @retval None
00332   */
00333 HAL_StatusTypeDef I2C1_WriteBuffer(uint16_t
Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuf
fer, uint16_t Length)
00334 {
00335   HAL_StatusTypeDef status = HAL_OK;
00336
00337   status = HAL_I2C_Mem_Write(&heval_I2c1, Ad
dr, Reg, RegSize, pBuffer, Length, I2c1Timeout);
00338
00339   /* Check the communication status */
00340   if(status != HAL_OK)
00341   {
00342     /* Re-Initiaize the BUS */
00343     I2C1_Error();
00344   }
00345   return status;
00346 }
00347
00348 /**
00349   * @brief  Manages error callback by re-ini
tializing I2C.
00350   * @retval None
00351   */
00352 void I2C1_Error(void)
```

```
00353 {
00354   /* De-initialize the I2C communication BUS
 */
00355   HAL_I2C_DeInit(&heval_I2c1);
00356
00357   /* Re-Initiaize the I2C communication BUS
*/
00358   I2C1_Init();
00359 }
00360
00361 /**
00362   * @brief I2C MSP Initialization
00363   * @param hi2c: I2C handle
00364   * @retval None
00365   */
00366 void I2C1_MspInit(I2C_HandleTypeDef *hi2c)
00367 {
00368   GPIO_InitTypeDef GPIO_InitStruct;
00369   RCC_PeriphCLKInitTypeDef RCC_PeriphCLKInit
Struct;
00370
00371   /*##-1- Set source clock to SYSCLK for I2C
1 ############################################
*/
00372   RCC_PeriphCLKInitStruct.PeriphClockSelecti
on = RCC_PERIPHCLK_I2C1;
00373   RCC_PeriphCLKInitStruct.I2c1ClockSelection
 = RCC_I2C1CLKSOURCE_SYSCLK;
00374   HAL_RCCEx_PeriphCLKConfig(&RCC_PeriphCLKIn
itStruct);
00375
00376   /*##-2- Configure the GPIOs ##############
#################################*/
00377
00378   /* Enable GPIO clock */
00379   BSP_I2C1_GPIO_CLK_ENABLE();
00380
```

```
00381   /* Configure I2C SCL & SDA as alternate fu
nction  */
00382   GPIO_InitStruct.Pin       = (BSP_I2C1_SCL_
PIN| BSP_I2C1_SDA_PIN);
00383   GPIO_InitStruct.Mode      = GPIO_MODE_AF_O
D;
00384   GPIO_InitStruct.Pull      = GPIO_NOPULL;
00385   GPIO_InitStruct.Speed     = GPIO_SPEED_FRE
Q_VERY_HIGH;
00386   GPIO_InitStruct.Alternate = BSP_I2C1_SCL_S
DA_AF;
00387   HAL_GPIO_Init(BSP_I2C1_GPIO_PORT, &GPIO_In
itStruct);
00388
00389   /*##-3- Configure the Eval I2C peripheral
######################################*/
00390   /* Enable I2C clock */
00391   BSP_I2C1_CLK_ENABLE();
00392
00393   /* Force the I2C peripheral clock reset */
00394   BSP_I2C1_FORCE_RESET();
00395
00396   /* Release the I2C peripheral clock reset
*/
00397   BSP_I2C1_RELEASE_RESET();
00398 }
00399
00400 #endif /*HAL_I2C_MODULE_ENABLED*/
00401
00402 #ifdef HAL_SPI_MODULE_ENABLED
00403 /**
00404   * @brief  Initializes SPI MSP.
00405   * @param  hspi: SPI handle
00406   * @retval None
00407   */
00408 static void SPIx_MspInit(SPI_HandleTypeDef *
hspi)
```

```
00409 {
00410   GPIO_InitTypeDef  GPIO_InitStruct;
00411
00412   /*** Configure the GPIOs ***/
00413   /* Enable GPIO clock */
00414   NUCLEO_SPIx_SCK_GPIO_CLK_ENABLE();
00415   NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_ENABLE();
00416
00417   /* Configure SPI SCK */
00418   GPIO_InitStruct.Pin = NUCLEO_SPIx_SCK_PIN;
00419   GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00420   GPIO_InitStruct.Pull  = GPIO_PULLUP;
00421   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VE
RY_HIGH;
00422   GPIO_InitStruct.Alternate = NUCLEO_SPIx_SC
K_AF;
00423   HAL_GPIO_Init(NUCLEO_SPIx_SCK_GPIO_PORT, &
GPIO_InitStruct);
00424
00425   /* Configure SPI MISO and MOSI */
00426   GPIO_InitStruct.Pin = NUCLEO_SPIx_MOSI_PIN
;
00427   GPIO_InitStruct.Alternate = NUCLEO_SPIx_MI
SO_MOSI_AF;
00428   GPIO_InitStruct.Pull  = GPIO_PULLDOWN;
00429   HAL_GPIO_Init(NUCLEO_SPIx_MISO_MOSI_GPIO_P
ORT, &GPIO_InitStruct);
00430
00431   GPIO_InitStruct.Pin = NUCLEO_SPIx_MISO_PIN
;
00432   HAL_GPIO_Init(NUCLEO_SPIx_MISO_MOSI_GPIO_P
ORT, &GPIO_InitStruct);
00433
00434   /*** Configure the SPI peripheral ***/
00435   /* Enable SPI clock */
00436   NUCLEO_SPIx_CLK_ENABLE();
00437 }
```

```
00438
00439 /**
00440   * @brief  Initializes SPI HAL.
00441   * @retval None
00442   */
00443 static void SPIx_Init(void)
00444 {
00445   if(HAL_SPI_GetState(&hnucleo_Spi) == HAL_S
PI_STATE_RESET)
00446   {
00447     /* SPI Config */
00448     hnucleo_Spi.Instance = NUCLEO_SPIx;
00449       /* SPI baudrate is set to 8 MHz maximu
m (PCLK2/SPI_BaudRatePrescaler = 32/4 = 8 MHz)
00450        to verify these constraints:
00451           - ST7735 LCD SPI interface max bau
drate is 15MHz for write and 6.66MHz for read
00452             Since the provided driver doesn'
t use read capability from LCD, only constraint
00453             on write baudrate is considered.
00454           - SD card SPI interface max baudra
te is 25MHz for write/read
00455           - PCLK2 max frequency is 32 MHz
00456        */
00457     hnucleo_Spi.Init.BaudRatePrescaler = SPI
_BAUDRATEPRESCALER_4;
00458     hnucleo_Spi.Init.Direction = SPI_DIRECTI
ON_2LINES;
00459     hnucleo_Spi.Init.CLKPhase = SPI_PHASE_2E
DGE;
00460     hnucleo_Spi.Init.CLKPolarity = SPI_POLAR
ITY_HIGH;
00461     hnucleo_Spi.Init.CRCCalculation = SPI_CR
CCALCULATION_DISABLE;
00462     hnucleo_Spi.Init.CRCPolynomial = 7;
00463     hnucleo_Spi.Init.DataSize = SPI_DATASIZE
_8BIT;
```

```
00464     hnucleo_Spi.Init.FirstBit = SPI_FIRSTBIT
_MSB;
00465     hnucleo_Spi.Init.NSS = SPI_NSS_SOFT;
00466     hnucleo_Spi.Init.TIMode = SPI_TIMODE_DIS
ABLE;
00467     hnucleo_Spi.Init.Mode = SPI_MODE_MASTER;
00468
00469     SPIx_MspInit(&hnucleo_Spi);
00470     HAL_SPI_Init(&hnucleo_Spi);
00471   }
00472 }
00473
00474 /**
00475   * @brief  SPI Read 4 bytes from device
00476   * @retval Read data
00477 */
00478 static uint32_t SPIx_Read(void)
00479 {
00480   HAL_StatusTypeDef status = HAL_OK;
00481   uint32_t readvalue = 0;
00482   uint32_t writevalue = 0xFFFFFFFF;
00483
00484   status = HAL_SPI_TransmitReceive(&hnucleo_
Spi, (uint8_t*) &writevalue, (uint8_t*) &readvalue
, 1, SpixTimeout);
00485
00486   /* Check the communication status */
00487   if(status != HAL_OK)
00488   {
00489     /* Execute user timeout callback */
00490     SPIx_Error();
00491   }
00492
00493   return readvalue;
00494 }
00495
00496 /**
```

```
00497   * @brief  SPI Write a byte to device
00498   * @param  Value: value to be written
00499   * @retval None
00500   */
00501 static void SPIx_Write(uint8_t Value)
00502 {
00503   HAL_StatusTypeDef status = HAL_OK;
00504
00505   status = HAL_SPI_Transmit(&hnucleo_Spi, (u
int8_t*) &Value, 1, SpixTimeout);
00506
00507   /* Check the communication status */
00508   if(status != HAL_OK)
00509   {
00510     /* Execute user timeout callback */
00511     SPIx_Error();
00512   }
00513 }
00514
00515 /**
00516   * @brief  SPI error treatment function
00517   * @retval None
00518   */
00519 static void SPIx_Error (void)
00520 {
00521   /* De-initialize the SPI communication BUS
 */
00522   HAL_SPI_DeInit(&hnucleo_Spi);
00523
00524   /* Re-Initiaize the SPI communication BUS
*/
00525   SPIx_Init();
00526 }
00527
00528 /*************************************************
*********************************
00529                                 LINK OPERATIONS
```

```
00530 ****************************************************
**********************************/
00531
00532 /********************************* LINK SD *
**********************************/
00533 /**
00534   * @brief  Initializes the SD Card and put
it into StandBy State (Ready for
00535   *         data transfer).
00536   * @retval None
00537   */
00538 void SD_IO_Init(void)
00539 {
00540   GPIO_InitTypeDef  GPIO_InitStruct;
00541   uint8_t counter;
00542
00543   /* SD_CS_GPIO Periph clock enable */
00544   SD_CS_GPIO_CLK_ENABLE();
00545
00546   /* Configure SD_CS_PIN pin: SD Card CS pin
 */
00547   GPIO_InitStruct.Pin = SD_CS_PIN;
00548   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP
;
00549   GPIO_InitStruct.Pull = GPIO_PULLUP;
00550   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VE
RY_HIGH;
00551   HAL_GPIO_Init(SD_CS_GPIO_PORT, &GPIO_InitS
truct);
00552
00553   /*------------Put SD in SPI mode----------
----*/
00554   /* SD SPI Config */
00555   SPIx_Init();
00556
00557   /* SD chip select high */
00558   SD_CS_HIGH();
```

```
00559
00560   /* Send dummy byte 0xFF, 10 times with CS
high */
00561   /* Rise CS and MOSI for 80 clocks cycles */

00562   for (counter = 0; counter <= 9; counter++)
00563   {
00564     /* Send dummy byte 0xFF */
00565     SD_IO_WriteByte(SD_DUMMY_BYTE);
00566   }
00567 }
00568
00569 /**
00570   * @brief  Writes a byte on the SD.
00571   * @param  Data: byte to send.
00572   * @retval None
00573   */
00574 void SD_IO_WriteByte(uint8_t Data)
00575 {
00576   /* Send the byte */
00577   SPIx_Write(Data);
00578 }
00579
00580 /**
00581   * @brief  Reads a byte from the SD.
00582   * @retval The received byte.
00583   */
00584 uint8_t SD_IO_ReadByte(void)
00585 {
00586   uint8_t data = 0;
00587
00588   /* Get the received data */
00589   data = SPIx_Read();
00590
00591   /* Return the shifted data */
00592   return data;
00593 }
```

```
00594
00595 /**
00596   * @brief  Sends 5 bytes command to the SD
card and get response
00597   * @param  Cmd: The user expected command t
o send to SD card.
00598   * @param  Arg: The command argument.
00599   * @param  Crc: The CRC.
00600   * @param  Response: Expected response from
 the SD card
00601   * @retval HAL_StatusTypeDef HAL Status
00602   */
00603 HAL_StatusTypeDef SD_IO_WriteCmd(uint8_t Cmd
, uint32_t Arg, uint8_t Crc, uint8_t Response)
00604 {
00605   uint32_t counter = 0x00;
00606   uint8_t frame[6];
00607
00608   /* Prepare Frame to send */
00609   frame[0] = (Cmd | 0x40);         /* Constr
uct byte 1 */
00610   frame[1] = (uint8_t)(Arg >> 24); /* Constr
uct byte 2 */
00611   frame[2] = (uint8_t)(Arg >> 16); /* Constr
uct byte 3 */
00612   frame[3] = (uint8_t)(Arg >> 8);  /* Constr
uct byte 4 */
00613   frame[4] = (uint8_t)(Arg);       /* Constr
uct byte 5 */
00614   frame[5] = (Crc);                /* Constr
uct byte 6 */
00615
00616   /* SD chip select low */
00617   SD_CS_LOW();
00618
00619   /* Send Frame */
00620   for (counter = 0; counter < 6; counter++)
```

```
00621   {
00622     SD_IO_WriteByte(frame[counter]); /* Send
 the Cmd bytes */
00623   }
00624
00625   if(Response != SD_NO_RESPONSE_EXPECTED)
00626   {
00627     return SD_IO_WaitResponse(Response);
00628   }
00629
00630   return HAL_OK;
00631 }
00632
00633 /**
00634   * @brief  Waits response from the SD card
00635   * @param  Response: Expected response from
 the SD card
00636   * @retval HAL_StatusTypeDef HAL Status
00637   */
00638 HAL_StatusTypeDef SD_IO_WaitResponse(uint8_t
 Response)
00639 {
00640   uint32_t timeout = 0xFFFF;
00641
00642   /* Check if response is got or a timeout i
s happen */
00643   while ((SD_IO_ReadByte() != Response) && t
imeout)
00644   {
00645     timeout--;
00646   }
00647
00648   if (timeout == 0)
00649   {
00650     /* After time out */
00651     return HAL_TIMEOUT;
00652   }
```

```
00653   else
00654   {
00655     /* Right response got */
00656     return HAL_OK;
00657   }
00658 }
00659
00660 /**
00661   * @brief  Sends dummy byte with CS High
00662   * @retval None
00663   */
00664 void SD_IO_WriteDummy(void)
00665 {
00666   /* SD chip select high */
00667   SD_CS_HIGH();
00668
00669   /* Send Dummy byte 0xFF */
00670   SD_IO_WriteByte(SD_DUMMY_BYTE);
00671 }
00672
00673 /****************************** LINK LCD
*******************************/
00674 /**
00675   * @brief  Initializes the LCD
00676   * @retval None
00677   */
00678 void LCD_IO_Init(void)
00679 {
00680   GPIO_InitTypeDef  GPIO_InitStruct;
00681
00682   /* LCD_CS_GPIO and LCD_DC_GPIO Periph cloc
k enable */
00683   LCD_CS_GPIO_CLK_ENABLE();
00684   LCD_DC_GPIO_CLK_ENABLE();
00685
00686   /* Configure LCD_CS_PIN pin: LCD Card CS p
in */
```

```
00687   GPIO_InitStruct.Pin = LCD_CS_PIN;
00688   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP
;
00689   GPIO_InitStruct.Pull = GPIO_NOPULL;
00690   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VE
RY_HIGH;
00691   HAL_GPIO_Init(LCD_CS_GPIO_PORT, &GPIO_Init
Struct);
00692
00693   /* Configure LCD_DC_PIN pin: LCD Card DC p
in */
00694   GPIO_InitStruct.Pin = LCD_DC_PIN;
00695   HAL_GPIO_Init(LCD_DC_GPIO_PORT, &GPIO_Init
Struct);
00696
00697   /* LCD chip select high */
00698   LCD_CS_HIGH();
00699
00700   /* LCD SPI Config */
00701   SPIx_Init();
00702 }
00703
00704 /**
00705   * @brief  Writes command to select the LCD
 register.
00706   * @param  LCDReg: Address of the selected
register.
00707   * @retval None
00708   */
00709 void LCD_IO_WriteReg(uint8_t LCDReg)
00710 {
00711   /* Reset LCD control line CS */
00712   LCD_CS_LOW();
00713
00714   /* Set LCD data/command line DC to Low */
00715   LCD_DC_LOW();
00716
```

```
00717   /* Send Command */
00718   SPIx_Write(LCDReg);
00719
00720   /* Deselect : Chip Select high */
00721   LCD_CS_HIGH();
00722 }
00723
00724 /**
00725   * @brief  Writes data to select the LCD re
gister.
00726   *         This function must be used after
 st7735_WriteReg() function
00727   * @param  Data: data to write to the selec
ted register.
00728   * @retval None
00729   */
00730 void LCD_IO_WriteData(uint8_t Data)
00731 {
00732   /* Reset LCD control line CS */
00733   LCD_CS_LOW();
00734
00735   /* Set LCD data/command line DC to High */
00736   LCD_DC_HIGH();
00737
00738   /* Send Data */
00739   SPIx_Write(Data);
00740
00741   /* Deselect : Chip Select high */
00742   LCD_CS_HIGH();
00743 }
00744
00745 /**
00746 * @brief  Write register value.
00747 * @param  pData Pointer on the register value

00748 * @param  Size Size of byte to transmit to t
he register
```

```
00749 * @retval None
00750 */
00751 void LCD_IO_WriteMultipleData(uint8_t *pData
, uint32_t Size)
00752 {
00753   uint32_t counter = 0;
00754
00755   /* Reset LCD control line CS */
00756   LCD_CS_LOW();
00757
00758   /* Set LCD data/command line DC to High */
00759   LCD_DC_HIGH();
00760
00761   if (Size == 1)
00762   {
00763     /* Only 1 byte to be sent to LCD - gener
al interface can be used */
00764     /* Send Data */
00765     SPIx_Write(*pData);
00766   }
00767   else
00768   {
00769     /* Several data should be sent in a raw
*/
00770     /* Direct SPI accesses for optimization
*/
00771     for (counter = Size; counter != 0; count
er--)
00772     {
00773       while(((hnucleo_Spi.Instance->SR) & SP
I_FLAG_TXE) != SPI_FLAG_TXE)
00774       {
00775       }
00776       /* Need to invert bytes for LCD*/
00777       *((__IO uint8_t*)&hnucleo_Spi.Instance
->DR) = *(pData+1);
00778
```

```
00779       while(((hnucleo_Spi.Instance->SR) & SP
I_FLAG_TXE) != SPI_FLAG_TXE)
00780       {
00781       }
00782       *((__IO uint8_t*)&hnucleo_Spi.Instance
->DR) = *pData;
00783       counter--;
00784       pData += 2;
00785     }
00786
00787     /* Wait until the bus is ready before re
leasing Chip select */
00788     while(((hnucleo_Spi.Instance->SR) & SPI_
FLAG_BSY) != RESET)
00789     {
00790     }
00791   }
00792   /* Deselect : Chip Select high */
00793   LCD_CS_HIGH();
00794 }
00795
00796 /**
00797   * @brief  Wait for loop in ms.
00798   * @param  Delay in ms.
00799   * @retval None
00800   */
00801 void LCD_Delay(uint32_t Delay)
00802 {
00803   HAL_Delay(Delay);
00804 }
00805 #endif /* HAL_SPI_MODULE_ENABLED */
00806
00807 /******************************* LINK JOYSTI
CK ********************************/
00808 #ifdef HAL_ADC_MODULE_ENABLED
00809 /**
00810   * @brief  Initializes ADC MSP.
```

```
00811   * @param  hadc: ADC peripheral
00812   * @retval None
00813   */
00814 static void ADCx_MspInit(ADC_HandleTypeDef *
hadc)
00815 {
00816   GPIO_InitTypeDef  GPIO_InitStruct;
00817
00818   /*** Configure the GPIOs ***/
00819   /* Enable GPIO clock */
00820   NUCLEO_ADCx_GPIO_CLK_ENABLE();
00821
00822   /* Configure ADC1 Channel8 as analog input
 */
00823   GPIO_InitStruct.Pin = NUCLEO_ADCx_GPIO_PIN
 ;
00824   GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
00825   HAL_GPIO_Init(NUCLEO_ADCx_GPIO_PORT, &GPIO
_InitStruct);
00826
00827   /*** Configure the ADC peripheral ***/
00828   /* Enable ADC clock */
00829   NUCLEO_ADCx_CLK_ENABLE();
00830 }
00831
00832 /**
00833   * @brief  Initializes ADC HAL.
00834   * @retval None
00835   */
00836 static void ADCx_Init(void)
00837 {
00838   if(HAL_ADC_GetState(&hnucleo_Adc) == HAL_A
DC_STATE_RESET)
00839   {
00840     /* ADC Config */
00841     hnucleo_Adc.Instance = NUCLEO_ADCx;
00842     hnucleo_Adc.Init.OversamplingMode      =
```

```
 DISABLE;
00843     hnucleo_Adc.Init.ClockPrescaler        =
 ADC_CLOCK_SYNC_PCLK_DIV2; /* (must not exceed 16M
Hz) */
00844     hnucleo_Adc.Init.LowPowerAutoPowerOff  =
 DISABLE;
00845     hnucleo_Adc.Init.LowPowerFrequencyMode =
 ENABLE;
00846     hnucleo_Adc.Init.LowPowerAutoWait      =
 ENABLE;
00847     hnucleo_Adc.Init.Resolution            =
 ADC_RESOLUTION_12B;
00848     hnucleo_Adc.Init.SamplingTime          =
 ADC_SAMPLETIME_1CYCLE_5;
00849     hnucleo_Adc.Init.ScanConvMode          =
 ADC_SCAN_DIRECTION_FORWARD;
00850     hnucleo_Adc.Init.DataAlign             =
 ADC_DATAALIGN_RIGHT;
00851     hnucleo_Adc.Init.ContinuousConvMode    =
 DISABLE;
00852     hnucleo_Adc.Init.DiscontinuousConvMode =
 DISABLE;
00853     hnucleo_Adc.Init.ExternalTrigConvEdge  =
 ADC_EXTERNALTRIGCONVEDGE_NONE;
00854     hnucleo_Adc.Init.EOCSelection          =
 ADC_EOC_SINGLE_CONV;
00855     hnucleo_Adc.Init.DMAContinuousRequests =
 DISABLE;
00856
00857     ADCx_MspInit(&hnucleo_Adc);
00858     HAL_ADC_Init(&hnucleo_Adc);
00859   }
00860 }
00861
00862 /**
00863   * @brief  Configures joystick available on
 adafruit 1.8" TFT shield
```

```
00864   *         managed through ADC to detect mo
tion.
00865   * @retval Joystickstatus (0=> success, 1=>
 fail)
00866   */
00867 uint8_t BSP_JOY_Init(void)
00868 {
00869   uint8_t status = 1;
00870
00871   ADCx_Init();
00872
00873   /* Start ADC calibration */
00874   HAL_ADCEx_Calibration_Start(&hnucleo_Adc,
ADC_SINGLE_ENDED);
00875
00876   /* Select Channel 0 to be converted */
00877   sConfig.Channel = ADC_CHANNEL_8;
00878   status = HAL_ADC_ConfigChannel(&hnucleo_Adc
, &sConfig);
00879
00880   /* Return Joystick initialization status */

00881   return status;
00882 }
00883
00884 /**
00885   * @brief  Returns the Joystick key pressed.

00886   * @note   To know which Joystick key is pr
essed we need to detect the voltage
00887   *         level on each key output
00888   *           - None  : 3.3 V / 4095
00889   *           - SEL   : 1.055 V / 1308
00890   *           - DOWN  : 0.71 V / 88
00891   *           - LEFT  : 3.0 V / 3720
00892   *           - RIGHT : 0.595 V / 737
00893   *           - UP    : 1.65 V / 2046
```

```
00894   * @retval JOYState_TypeDef: Code of the Joystick key pressed.
00895   */
00896 JOYState_TypeDef BSP_JOY_GetState(void)
00897 {
00898   JOYState_TypeDef state;
00899   uint16_t  KeyConvertedValue = 0;
00900
00901  /* Start the conversion process */
00902   HAL_ADC_Start(&hnucleo_Adc);
00903
00904   /* Wait for the end of conversion */
00905   if (HAL_ADC_PollForConversion(&hnucleo_Adc, 10) != HAL_TIMEOUT)
00906   {
00907     /* Get the converted value of regular channel */
00908     KeyConvertedValue = HAL_ADC_GetValue(&hnucleo_Adc);
00909   }
00910
00911   if((KeyConvertedValue > 2010) && (KeyConvertedValue < 2090))
00912   {
00913     state = JOY_UP;
00914   }
00915   else if((KeyConvertedValue > 680) && (KeyConvertedValue < 780))
00916   {
00917     state = JOY_RIGHT;
00918   }
00919   else if((KeyConvertedValue > 1270) && (KeyConvertedValue < 1350))
00920   {
00921     state = JOY_SEL;
00922   }
00923   else if((KeyConvertedValue > 50) && (KeyCo
```

```
nvertedValue < 130))
00924   {
00925     state = JOY_DOWN;
00926   }
00927   else if((KeyConvertedValue > 3570) && (Key
ConvertedValue < 3800))
00928   {
00929     state = JOY_LEFT;
00930   }
00931   else
00932   {
00933     state = JOY_NONE;
00934   }
00935
00936   /* Loop while a key is pressed */
00937   if(state != JOY_NONE)
00938   {
00939     KeyConvertedValue = HAL_ADC_GetValue(&hn
ucleo_Adc);
00940   }
00941   /* Return the code of the Joystick key pre
ssed */
00942   return state;
00943 }
00944 #endif /* HAL_ADC_MODULE_ENABLED */
00945
00946 /**
00947   * @}
00948   */
00949
00950 /**
00951   * @}
00952   */
00953
00954 /**
00955   * @}
00956   */
```

```
00957
00958 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM32L0xx_Nucleo_32 BSP User Manual

BSP

Modules

NUCLEO 32

This file provides set of firmware functions to manage Leds and push-button available on STM32L0XX-Nucleo Kit from STMicroelectronics.