STM32L0xx_Nucleo BSP User Manual

Private Types Definitions

STM32L0XX_NUCLEO_LOW_LEVEL

Generated on Mon Aug 28 2017 14:45:03 for STM32L0xx_Nucleo BSP User Manual by doxygen 1.7.6.1

STM32L0xx_Nucleo BSP User Manual

Exported Macros

Exported Constants

Generated on Mon Aug 28 2017 14:45:03 for STM32L0xx_Nucleo BSP User Manual by doxygen 1.7.6.1

STM32L0xx_Nucleo 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_BSP_VERSION : **stm32l0xx_nucleo.c**
- __STM32L0XX_NUCLEO_BSP_VERSION_MAIN : **stm32l0xx_nucleo.c**
- __STM32L0XX_NUCLEO_BSP_VERSION_RC : **stm32l0xx_nucleo.c**
- __STM32L0XX_NUCLEO_BSP_VERSION_SUB1 : **stm32l0xx_nucleo.c**
- __STM32L0XX_NUCLEO_BSP_VERSION_SUB2 : **stm32l0xx_nucleo.c**

- a -

- ADCx_DeInit() : **stm32l0xx_nucleo.c**
- ADCx_Init() : **stm32l0xx_nucleo.c**
- ADCx_MspDeInit() : **stm32l0xx_nucleo.c**
- ADCx_MspInit() : **stm32l0xx_nucleo.c**

- b -

- BSP_GetVersion() : **stm32l0xx_nucleo.c** , **stm32l0xx_nucleo.h**
- BSP_JOY_DeInit() : **stm32l0xx_nucleo.h** , **stm32l0xx_nucleo.c**
- BSP_JOY_GetState() : **stm32l0xx_nucleo.c** ,

stm32l0xx_nucleo.h

- BSP_JOY_Init() : **stm32l0xx_nucleo.h** , **stm32l0xx_nucleo.c**
- BSP_LED_DeInit() : **stm32l0xx_nucleo.c** , **stm32l0xx_nucleo.h**
- BSP_LED_Init() : **stm32l0xx_nucleo.c** , **stm32l0xx_nucleo.h**
- BSP_LED_Off() : **stm32l0xx_nucleo.c** , **stm32l0xx_nucleo.h**
- BSP_LED_On() : **stm32l0xx_nucleo.h** , **stm32l0xx_nucleo.c**
- BSP_LED_Toggle() : **stm32l0xx_nucleo.c** , **stm32l0xx_nucleo.h**
- BSP_PB_DeInit() : **stm32l0xx_nucleo.c** , **stm32l0xx_nucleo.h**
- BSP_PB_GetState() : **stm32l0xx_nucleo.c** , **stm32l0xx_nucleo.h**
- BSP_PB_Init() : **stm32l0xx_nucleo.c** , **stm32l0xx_nucleo.h**
- BUTTON_IRQn : **stm32l0xx_nucleo.c**
- BUTTON_KEY : **stm32l0xx_nucleo.h**
- BUTTON_MODE_EXTI : **stm32l0xx_nucleo.h**
- BUTTON_MODE_GPIO : **stm32l0xx_nucleo.h**
- BUTTON_PIN : **stm32l0xx_nucleo.c**
- BUTTON_PORT : **stm32l0xx_nucleo.c**
- Button_TypeDef : **stm32l0xx_nucleo.h**
- BUTTON_USER : **stm32l0xx_nucleo.h**
- ButtonMode_TypeDef : **stm32l0xx_nucleo.h**
- BUTTONn : **stm32l0xx_nucleo.h**
- BUTTONx_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- BUTTONx_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**

- h -

- hnucleo_Adc : **stm32l0xx_nucleo.c**
- hnucleo_Spi : **stm32l0xx_nucleo.c**

- j -

- JOY_DOWN : **stm32l0xx_nucleo.h**
- JOY_LEFT : **stm32l0xx_nucleo.h**
- JOY_NONE : **stm32l0xx_nucleo.h**
- JOY_RIGHT : **stm32l0xx_nucleo.h**
- JOY_SEL : **stm32l0xx_nucleo.h**
- JOY_UP : **stm32l0xx_nucleo.h**
- JOYState_TypeDef : **stm32l0xx_nucleo.h**

- k -

- KEY_BUTTON_EXTI_IRQn : **stm32l0xx_nucleo.h**
- KEY_BUTTON_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- KEY_BUTTON_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**
- KEY_BUTTON_GPIO_PORT : **stm32l0xx_nucleo.h**
- KEY_BUTTON_PIN : **stm32l0xx_nucleo.h**

- l -

- LCD_CS_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- LCD_CS_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**
- LCD_CS_GPIO_PORT : **stm32l0xx_nucleo.h**
- LCD_CS_HIGH : **stm32l0xx_nucleo.h**
- LCD_CS_LOW : **stm32l0xx_nucleo.h**
- LCD_CS_PIN : **stm32l0xx_nucleo.h**
- LCD_DC_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- LCD_DC_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**
- LCD_DC_GPIO_PORT : **stm32l0xx_nucleo.h**
- LCD_DC_HIGH : **stm32l0xx_nucleo.h**
- LCD_DC_LOW : **stm32l0xx_nucleo.h**
- LCD_DC_PIN : **stm32l0xx_nucleo.h**
- LCD_Delay() : **stm32l0xx_nucleo.c**
- LCD_IO_Init() : **stm32l0xx_nucleo.c**
- LCD_IO_WriteData() : **stm32l0xx_nucleo.c**
- LCD_IO_WriteMultipleData() : **stm32l0xx_nucleo.c**
- LCD_IO_WriteReg() : **stm32l0xx_nucleo.c**
- LED2 : **stm32l0xx_nucleo.h**
- LED2_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- LED2_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**
- LED2_GPIO_PORT : **stm32l0xx_nucleo.h**
- LED2_PIN : **stm32l0xx_nucleo.h**
- LED_GREEN : **stm32l0xx_nucleo.h**
- LED_PIN : **stm32l0xx_nucleo.c**
- LED_PORT : **stm32l0xx_nucleo.c**
- Led_TypeDef : **stm32l0xx_nucleo.h**
- LEDn : **stm32l0xx_nucleo.h**
- LEDx_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**

- LEDx_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**

- n -

- NUCLEO_ADCx : **stm32l0xx_nucleo.h**
- NUCLEO_ADCx_CLK_DISABLE : **stm32l0xx_nucleo.h**
- NUCLEO_ADCx_CLK_ENABLE : **stm32l0xx_nucleo.h**
- NUCLEO_ADCx_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- NUCLEO_ADCx_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**
- NUCLEO_ADCx_GPIO_PIN : **stm32l0xx_nucleo.h**
- NUCLEO_ADCx_GPIO_PORT : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_CLK_ENABLE : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_MISO_MOSI_AF : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_MISO_MOSI_GPIO_PORT : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_MISO_PIN : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_MOSI_PIN : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_SCK_AF : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_SCK_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_SCK_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_SCK_GPIO_PORT : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_SCK_PIN : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_TIMEOUT_MAX : **stm32l0xx_nucleo.h**

- s -

- sConfig : **stm32l0xx_nucleo.c**
- SD_CS_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- SD_CS_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**
- SD_CS_GPIO_PORT : **stm32l0xx_nucleo.h**
- SD_CS_HIGH : **stm32l0xx_nucleo.h**

- SD_CS_LOW : **stm32l0xx_nucleo.h**
- SD_CS_PIN : **stm32l0xx_nucleo.h**
- SD_DUMMY_BYTE : **stm32l0xx_nucleo.c**
- SD_IO_CSState() : **stm32l0xx_nucleo.c**
- SD_IO_Init() : **stm32l0xx_nucleo.c**
- SD_IO_ReadByte() : **stm32l0xx_nucleo.c**
- SD_IO_ReadData() : **stm32l0xx_nucleo.c**
- SD_IO_WriteByte() : **stm32l0xx_nucleo.c**
- SD_IO_WriteData() : **stm32l0xx_nucleo.c**
- SD_IO_WriteReadData() : **stm32l0xx_nucleo.c**
- SD_NO_RESPONSE_EXPECTED : **stm32l0xx_nucleo.c**
- SPIx_Error() : **stm32l0xx_nucleo.c**
- SPIx_Init() : **stm32l0xx_nucleo.c**
- SPIx_MspInit() : **stm32l0xx_nucleo.c**
- SPIx_Write() : **stm32l0xx_nucleo.c**
- SPIx_WriteData() : **stm32l0xx_nucleo.c**
- SPIx_WriteReadData() : **stm32l0xx_nucleo.c**
- SpixTimeout : **stm32l0xx_nucleo.c**

- u -

- USER_BUTTON_EXTI_IRQn : **stm32l0xx_nucleo.h**
- USER_BUTTON_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- USER_BUTTON_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**
- USER_BUTTON_GPIO_PORT : **stm32l0xx_nucleo.h**
- USER_BUTTON_PIN : **stm32l0xx_nucleo.h**

STM32L0xx_Nucleo BSP User Manual

- a -

- ADCx_DeInit() : **stm32l0xx_nucleo.c**
- ADCx_Init() : **stm32l0xx_nucleo.c**
- ADCx_MspDeInit() : **stm32l0xx_nucleo.c**
- ADCx_MspInit() : **stm32l0xx_nucleo.c**

- b -

- BSP_GetVersion() : **stm32l0xx_nucleo.c** , **stm32l0xx_nucleo.h**
- BSP_JOY_DeInit() : **stm32l0xx_nucleo.h** , **stm32l0xx_nucleo.c**
- BSP_JOY_GetState() : **stm32l0xx_nucleo.c** , **stm32l0xx_nucleo.h**
- BSP_JOY_Init() : **stm32l0xx_nucleo.h** , **stm32l0xx_nucleo.c**
- BSP_LED_DeInit() : **stm32l0xx_nucleo.c** , **stm32l0xx_nucleo.h**
- BSP_LED_Init() : **stm32l0xx_nucleo.c** , **stm32l0xx_nucleo.h**
- BSP_LED_Off() : **stm32l0xx_nucleo.c** , **stm32l0xx_nucleo.h**
- BSP_LED_On() : **stm32l0xx_nucleo.c** , **stm32l0xx_nucleo.h**
- BSP_LED_Toggle() : **stm32l0xx_nucleo.c** , **stm32l0xx_nucleo.h**
- BSP_PB_DeInit() : **stm32l0xx_nucleo.c** , **stm32l0xx_nucleo.h**
- BSP_PB_GetState() : **stm32l0xx_nucleo.c** , **stm32l0xx_nucleo.h**
- BSP_PB_Init() : **stm32l0xx_nucleo.h** , **stm32l0xx_nucleo.c**

- l -

- LCD_Delay() : **stm32l0xx_nucleo.c**
- LCD_IO_Init() : **stm32l0xx_nucleo.c**
- LCD_IO_WriteData() : **stm32l0xx_nucleo.c**
- LCD_IO_WriteMultipleData() : **stm32l0xx_nucleo.c**
- LCD_IO_WriteReg() : **stm32l0xx_nucleo.c**

- s -

- SD_IO_CSState() : **stm32l0xx_nucleo.c**
- SD_IO_Init() : **stm32l0xx_nucleo.c**
- SD_IO_ReadByte() : **stm32l0xx_nucleo.c**
- SD_IO_ReadData() : **stm32l0xx_nucleo.c**
- SD_IO_WriteByte() : **stm32l0xx_nucleo.c**
- SD_IO_WriteData() : **stm32l0xx_nucleo.c**
- SD_IO_WriteReadData() : **stm32l0xx_nucleo.c**
- SPIx_Error() : **stm32l0xx_nucleo.c**
- SPIx_Init() : **stm32l0xx_nucleo.c**
- SPIx_MspInit() : **stm32l0xx_nucleo.c**
- SPIx_Write() : **stm32l0xx_nucleo.c**
- SPIx_WriteData() : **stm32l0xx_nucleo.c**
- SPIx_WriteReadData() : **stm32l0xx_nucleo.c**

STM32L0xx_Nucleo BSP User Manual

- BUTTON_IRQn : **stm32l0xx_nucleo.c**
- BUTTON_PIN : **stm32l0xx_nucleo.c**
- BUTTON_PORT : **stm32l0xx_nucleo.c**
- hnucleo_Adc : **stm32l0xx_nucleo.c**
- hnucleo_Spi : **stm32l0xx_nucleo.c**
- LED_PIN : **stm32l0xx_nucleo.c**
- LED_PORT : **stm32l0xx_nucleo.c**
- sConfig : **stm32l0xx_nucleo.c**
- SpixTimeout : **stm32l0xx_nucleo.c**

STM32L0xx_Nucleo BSP User Manual

- Button_TypeDef : **stm32l0xx_nucleo.h**
- ButtonMode_TypeDef : **stm32l0xx_nucleo.h**
- JOYState_TypeDef : **stm32l0xx_nucleo.h**
- Led_TypeDef : **stm32l0xx_nucleo.h**

STM32L0xx_Nucleo BSP User Manual

- BUTTON_KEY : **stm32l0xx_nucleo.h**
- BUTTON_MODE_EXTI : **stm32l0xx_nucleo.h**
- BUTTON_MODE_GPIO : **stm32l0xx_nucleo.h**
- BUTTON_USER : **stm32l0xx_nucleo.h**
- JOY_DOWN : **stm32l0xx_nucleo.h**
- JOY_LEFT : **stm32l0xx_nucleo.h**
- JOY_NONE : **stm32l0xx_nucleo.h**
- JOY_RIGHT : **stm32l0xx_nucleo.h**
- JOY_SEL : **stm32l0xx_nucleo.h**
- JOY_UP : **stm32l0xx_nucleo.h**
- LED2 : **stm32l0xx_nucleo.h**
- LED_GREEN : **stm32l0xx_nucleo.h**

STM32L0xx_Nucleo BSP User Manual

- _ -

- __STM32L0XX_NUCLEO_BSP_VERSION : **stm32l0xx_nucleo.c**
- __STM32L0XX_NUCLEO_BSP_VERSION_MAIN : **stm32l0xx_nucleo.c**
- __STM32L0XX_NUCLEO_BSP_VERSION_RC : **stm32l0xx_nucleo.c**
- __STM32L0XX_NUCLEO_BSP_VERSION_SUB1 : **stm32l0xx_nucleo.c**
- __STM32L0XX_NUCLEO_BSP_VERSION_SUB2 : **stm32l0xx_nucleo.c**

- b -

- BUTTONn : **stm32l0xx_nucleo.h**
- BUTTONx_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- BUTTONx_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**

- k -

- KEY_BUTTON_EXTI_IRQn : **stm32l0xx_nucleo.h**
- KEY_BUTTON_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- KEY_BUTTON_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**
- KEY_BUTTON_GPIO_PORT : **stm32l0xx_nucleo.h**
- KEY_BUTTON_PIN : **stm32l0xx_nucleo.h**

- l -

- LCD_CS_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- LCD_CS_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**
- LCD_CS_GPIO_PORT : **stm32l0xx_nucleo.h**
- LCD_CS_HIGH : **stm32l0xx_nucleo.h**
- LCD_CS_LOW : **stm32l0xx_nucleo.h**
- LCD_CS_PIN : **stm32l0xx_nucleo.h**
- LCD_DC_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- LCD_DC_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**
- LCD_DC_GPIO_PORT : **stm32l0xx_nucleo.h**
- LCD_DC_HIGH : **stm32l0xx_nucleo.h**
- LCD_DC_LOW : **stm32l0xx_nucleo.h**
- LCD_DC_PIN : **stm32l0xx_nucleo.h**
- LED2_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- LED2_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**
- LED2_GPIO_PORT : **stm32l0xx_nucleo.h**
- LED2_PIN : **stm32l0xx_nucleo.h**
- LEDn : **stm32l0xx_nucleo.h**
- LEDx_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- LEDx_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**

- n -

- NUCLEO_ADCx : **stm32l0xx_nucleo.h**
- NUCLEO_ADCx_CLK_DISABLE : **stm32l0xx_nucleo.h**
- NUCLEO_ADCx_CLK_ENABLE : **stm32l0xx_nucleo.h**
- NUCLEO_ADCx_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- NUCLEO_ADCx_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**
- NUCLEO_ADCx_GPIO_PIN : **stm32l0xx_nucleo.h**
- NUCLEO_ADCx_GPIO_PORT : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_CLK_ENABLE : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_MISO_MOSI_AF : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**

- NUCLEO_SPIx_MISO_MOSI_GPIO_PORT : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_MISO_PIN : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_MOSI_PIN : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_SCK_AF : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_SCK_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_SCK_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_SCK_GPIO_PORT : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_SCK_PIN : **stm32l0xx_nucleo.h**
- NUCLEO_SPIx_TIMEOUT_MAX : **stm32l0xx_nucleo.h**

- s -

- SD_CS_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- SD_CS_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**
- SD_CS_GPIO_PORT : **stm32l0xx_nucleo.h**
- SD_CS_HIGH : **stm32l0xx_nucleo.h**
- SD_CS_LOW : **stm32l0xx_nucleo.h**
- SD_CS_PIN : **stm32l0xx_nucleo.h**
- SD_DUMMY_BYTE : **stm32l0xx_nucleo.c**
- SD_NO_RESPONSE_EXPECTED : **stm32l0xx_nucleo.c**

- u -

- USER_BUTTON_EXTI_IRQn : **stm32l0xx_nucleo.h**
- USER_BUTTON_GPIO_CLK_DISABLE : **stm32l0xx_nucleo.h**
- USER_BUTTON_GPIO_CLK_ENABLE : **stm32l0xx_nucleo.h**
- USER_BUTTON_GPIO_PORT : **stm32l0xx_nucleo.h**
- USER_BUTTON_PIN : **stm32l0xx_nucleo.h**

STM32L0xx_Nucleo BSP User Manual

stm32l0xx_nucleo.c File Reference

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

```
#include "stm32l0xx_nucleo.h"
```

Go to the source code of this file.

Defines

#define	**__STM32L0XX_NUCLEO_BSP_VERSION_MAIN** (0x02) STM32L0XX NUCLEO BSP Driver version number.
#define	**__STM32L0XX_NUCLEO_BSP_VERSION_SUB1** (0x01)
#define	**__STM32L0XX_NUCLEO_BSP_VERSION_SUB2** (0x01)
#define	**__STM32L0XX_NUCLEO_BSP_VERSION_RC** (0x00)
#define	**__STM32L0XX_NUCLEO_BSP_VERSION**
#define	**SD_DUMMY_BYTE** 0xFF LINK SD Card.
#define	**SD_NO_RESPONSE_EXPECTED** 0x80

Functions

static void	**SPIx_Init** (void) Initialize SPI HAL.
static void	**SPIx_Write** (uint8_t Value) SPI Write a byte to device.
static void	**SPIx_WriteData** (uint8_t *DataIn, uint16_t DataLength) SPI Write an amount of data to device.
static void	**SPIx_WriteReadData** (const uint8_t *DataIn, uint8_t *DataOut, uint16_t DataLength) SPI Write a byte to device.
static void	**SPIx_Error** (void) SPI error treatment function.
static void	**SPIx_MspInit** (void) Initialize SPI MSP.
void	**SD_IO_Init** (void) Initialize the SD Card and put it into StandBy State (Ready for data transfer).
void	**SD_IO_CSState** (uint8_t val) Set the SD_CS pin.
void	**SD_IO_WriteReadData** (const uint8_t *DataIn, uint8_t *DataOut, uint16_t DataLength) Write byte(s) on the SD.
void	**SD_IO_ReadData** (uint8_t *DataOut, uint16_t DataLength) Write an amount of data on the SD.
void	**SD_IO_WriteData** (const uint8_t *Data, uint16_t DataLength) Write an amount of data on the SD.
uint8_t	**SD_IO_WriteByte** (uint8_t Data)

	Write a byte on the SD.
uint8_t	**SD_IO_ReadByte** (void)
void	**LCD_IO_Init** (void) Initialize 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) Write command to select the LCD register.
void	**LCD_Delay** (uint32_t Delay) Wait for loop in ms.
static HAL_StatusTypeDef	**ADCx_Init** (void) Initializes ADC HAL.
static void	**ADCx_DeInit** (void) Initializes ADC HAL.
static void	**ADCx_MspInit** (ADC_HandleTypeDef *hadc) Initialize ADC MSP.
static void	**ADCx_MspDeInit** (ADC_HandleTypeDef *hadc) DeInitializes 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_DeInit** (**Led_TypeDef** Led) DeInit LEDs.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.

void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** ButtonMode) Configures Button GPIO and EXTI Line.
void	**BSP_PB_DeInit** (**Button_TypeDef** Button) Push Button DeInit.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected Button state.
uint8_t	**BSP_JOY_Init** (void) Configures joystick available on adafruit 1.8" TFT shield managed through ADC to detect motion.
void	**BSP_JOY_DeInit** (void) DeInit joystick GPIOs.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the Joystick key pressed.

Variables

GPIO_TypeDef *	**LED_PORT** [**LEDn**] = {**LED2_GPIO_PORT**}
const uint16_t	**LED_PIN** [**LEDn**] = {**LED2_PIN**}
GPIO_TypeDef *	**BUTTON_PORT** [**BUTTONn**] = {**USER_BUTTON_GPIO_PORT** }
const uint16_t	**BUTTON_PIN** [**BUTTONn**] = {**USER_BUTTON_PIN** }
const uint8_t	**BUTTON_IRQn** [**BUTTONn**] = {**USER_BUTTON_EXTI_IRQn** }
uint32_t	**SpixTimeout** = **NUCLEO_SPIx_TIMEOUT_MAX** BUS variables.
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
- LCD, joystick and microSD available on Adafruit 1.8" TFT LCD shield (reference ID 802)

Attention:

Definition in file **stm32l0xx_nucleo.c**.

STM32L0xx_Nucleo BSP User Manual

stm32l0xx_nucleo.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	**LED2_PIN** GPIO_PIN_5
#define	**LED2_GPIO_PORT** GPIOA
#define	**LED2_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOA_CLK_E
#define	**LED2_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOA_CLK_D
#define	**LEDx_GPIO_CLK_ENABLE**(__INDEX__) do { if((__INDEX_ **LED2_GPIO_CLK_ENABLE**();} while(0)
#define	**LEDx_GPIO_CLK_DISABLE**(__INDEX__) (((__INDEX__) == **LED2_GPIO_CLK_DISABLE**() : 0)
#define	**BUTTONn** 1
#define	**USER_BUTTON_PIN** GPIO_PIN_13 User push-button.
#define	**USER_BUTTON_GPIO_PORT** GPIOC
#define	**USER_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_GPIC
#define	**USER_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_GPI
#define	**USER_BUTTON_EXTI_IRQn** EXTI4_15_IRQn
#define	**KEY_BUTTON_PIN** **USER_BUTTON_PIN**
#define	**KEY_BUTTON_GPIO_PORT** **USER_BUTTON_GPIO_PORT**
#define	**KEY_BUTTON_GPIO_CLK_ENABLE**() **USER_BUTTON_GP**
#define	**KEY_BUTTON_GPIO_CLK_DISABLE**() **USER_BUTTON_G**
#define	**KEY_BUTTON_EXTI_IRQn** **USER_BUTTON_EXTI_IRQn**
#define	**BUTTONx_GPIO_CLK_ENABLE**(__INDEX__) do { if((__IND **USER_BUTTON_GPIO_CLK_ENABLE**();} while(0)
#define	**BUTTONx_GPIO_CLK_DISABLE**(__INDEX__) (((__INDEX_ **USER_BUTTON_GPIO_CLK_DISABLE**() : 0)
#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_5
#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** GPIOA

#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_6

#define **NUCLEO_SPIx_MOSI_PIN** GPIO_PIN_7

#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 (shield D4)

#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 (shield D10)

#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_CLK

#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_CLK
#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_CLK_DISABLE()  __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_GPIO
#define NUCLEO_ADCx_GPIO_CLK_DISABLE()  __HAL_RCC_GPI
```

Enumerations

enum	**Led_TypeDef** { **LED2** = 0, **LED_GREEN** = LED2 }
enum	**Button_TypeDef** { **BUTTON_USER** = 0, **BUTTON_KEY** = BUTTON_USER }
enum	**ButtonMode_TypeDef** { **BUTTON_MODE_GPIO** = 0, **BUTTON_MODE_EXTI** = 1 }

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

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_DeInit** (**Led_TypeDef** Led) DeInit LEDs.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** ButtonMode) Configures Button GPIO and EXTI Line.
void	**BSP_PB_DeInit** (**Button_TypeDef** Button) Push Button DeInit.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected Button state.
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.
void	**BSP_JOY_DeInit** (void) DeInit joystick GPIOs.

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:

Definition in file **stm32l0xx_nucleo.h**.

STM32L0xx_Nucleo BSP User Manual

Modules

Here is a list of all modules:

STM32L0xx_Nucleo BSP User Manual

File List

Here is a list of all files with brief descriptions:

stm32l0xx_nucleo.c [code]	This file provides set of firmware functions to manage:
stm32l0xx_nucleo.h [code]	This file contains definitions for:

STM32L0xx_Nucleo BSP User Manual

Directories

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

- Firmware
  - Drivers
    - BSP
      - STM32L0xx_Nucleo

Generated on Mon Aug 28 2017 14:45:03 for STM32L0xx_Nucleo BSP User Manual by doxygen 1.7.6.1

STM32L0xx_Nucleo BSP User Manual

STM32L0XX_NUCLEO_LOW_LEVEL

STM32L0XX_NUCLEO

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 BSP User Manual

Exported Constants

STM32L0XX_NUCLEO_LOW_LEVEL

Modules

Exported Macros
Exported Functions
STM32L0XX_NUCLEO_LOW_LEVEL_LED Define for STM32L0XX_NUCLEO board.
STM32L0XX_NUCLEO_LOW_LEVEL_BUTTON
STM32L0XX_NUCLEO_LOW_LEVEL_BUS
STM32L0XX_NUCLEO_LOW_LEVEL_COMPONENT

STM32L0xx_Nucleo BSP User Manual

Private Defines

STM32L0XX_NUCLEO_LOW_LEVEL

Defines

#define	**__STM32L0XX_NUCLEO_BSP_VERSION_MAIN** (0x02) STM32L0XX NUCLEO BSP Driver version number.
#define	**__STM32L0XX_NUCLEO_BSP_VERSION_SUB1** (0x01)
#define	**__STM32L0XX_NUCLEO_BSP_VERSION_SUB2** (0x01)
#define	**__STM32L0XX_NUCLEO_BSP_VERSION_RC** (0x00)
#define	**__STM32L0XX_NUCLEO_BSP_VERSION**
#define	**SD_DUMMY_BYTE** 0xFF LINK SD Card.
#define	**SD_NO_RESPONSE_EXPECTED** 0x80

Define Documentation

#define __STM32L0XX_NUCLEO_BSP_VERSION

Value:

```
((__STM32L0XX_NUCLEO_BSP_VERSION_MAIN << 24)\
                                                |(__
STM32L0XX_NUCLEO_BSP_VERSION_SUB1 << 16)\
                                                |(__
STM32L0XX_NUCLEO_BSP_VERSION_SUB2 << 8 )\
                                                |(__
STM32L0XX_NUCLEO_BSP_VERSION_RC))
```

Definition at line **76** of file **stm32l0xx_nucleo.c**.

Referenced by **BSP_GetVersion()**.

#define __STM32L0XX_NUCLEO_BSP_VERSION_MAIN (0x02)

STM32L0XX NUCLEO BSP Driver version number.

[31:24] main version

Definition at line **72** of file **stm32l0xx_nucleo.c**.

#define __STM32L0XX_NUCLEO_BSP_VERSION_RC (0x00)

[7:0] release candidate

Definition at line **75** of file **stm32l0xx_nucleo.c**.

#define __STM32L0XX_NUCLEO_BSP_VERSION_SUB1 (0x01)

[23:16] sub1 version

Definition at line **73** of file **stm32l0xx_nucleo.c**.

#define __STM32L0XX_NUCLEO_BSP_VERSION_SUB2 (0x01)

[15:8] sub2 version

Definition at line **74** of file **stm32l0xx_nucleo.c**.

#define SD_DUMMY_BYTE 0xFF

LINK SD Card.

Definition at line **84** of file **stm32l0xx_nucleo.c**.

Referenced by **SD_IO_Init()**.

#define SD_NO_RESPONSE_EXPECTED 0x80

Definition at line **85** of file **stm32l0xx_nucleo.c**.

STM32L0xx_Nucleo BSP User Manual

Private Functions

STM32L0XX_NUCLEO_LOW_LEVEL

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_DeInit** (**Led_TypeDef** Led) DeInit LEDs.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** ButtonMode) Configures Button GPIO and EXTI Line.
void	**BSP_PB_DeInit** (**Button_TypeDef** Button) Push Button DeInit.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected Button state.
uint8_t	**BSP_JOY_Init** (void) Configures joystick available on adafruit 1.8" TFT shield managed through ADC to detect motion.
void	**BSP_JOY_DeInit** (void) DeInit joystick GPIOs.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the Joystick key pressed.
static void	**SPIx_MspInit** (void)

	Initialize SPI MSP.
static void	**SPIx_Init** (void) Initialize SPI HAL.
static void	**SPIx_WriteReadData** (const uint8_t *DataIn, uint8_t *DataOut, uint16_t DataLength) SPI Write a byte to device.
static void	**SPIx_WriteData** (uint8_t *DataIn, uint16_t DataLength) SPI Write an amount of data to 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) Initialize the SD Card and put it into StandBy State (Ready for data transfer).
void	**SD_IO_CSState** (uint8_t val) Set the SD_CS pin.
void	**SD_IO_WriteReadData** (const uint8_t *DataIn, uint8_t *DataOut, uint16_t DataLength) Write byte(s) on the SD.
uint8_t	**SD_IO_WriteByte** (uint8_t Data) Write a byte on the SD.
void	**SD_IO_ReadData** (uint8_t *DataOut, uint16_t DataLength) Write an amount of data on the SD.
void	**SD_IO_WriteData** (const uint8_t *Data, uint16_t DataLength) Write an amount of data on the SD.
void	**LCD_IO_Init** (void) Initialize the LCD.
void	**LCD_IO_WriteReg** (uint8_t LCDReg)

	Write 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) Initialize ADC MSP.
static void	**ADCx_MspDeInit** (ADC_HandleTypeDef *hadc) DeInitializes ADC MSP.
static HAL_StatusTypeDef	**ADCx_Init** (void) Initializes ADC HAL.
static void	**ADCx_DeInit** (void) Initializes ADC HAL.

Function Documentation

static void ADCx_DeInit (void) [static]

Initializes ADC HAL.

Return values:
None

Definition at line **826** of file **stm32l0xx_nucleo.c**.

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

Referenced by **BSP_JOY_DeInit()**.

static HAL_StatusTypeDef ADCx_Init (void) [static]

Initializes ADC HAL.

Return values:
None

Definition at line **779** of file **stm32l0xx_nucleo.c**.

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

Referenced by **BSP_JOY_Init()**.

static void ADCx_MspDeInit (ADC_HandleTypeDef * hadc) [static

DeInitializes ADC MSP.

Parameters:

hadc,: ADC peripheral

Note:
ADC DeInit does not disable the GPIO clock

Return values:
None

Definition at line **759** of file **stm32l0xx_nucleo.c**.

References **NUCLEO_ADCx_CLK_DISABLE**, **NUCLEO_ADCx_GPIO_PIN**, and **NUCLEO_ADCx_GPIO_PORT**.

Referenced by **ADCx_DeInit()**.

static void ADCx_MspInit (ADC_HandleTypeDef * hadc) [static]

Initialize ADC MSP.

Return values:
None

Definition at line **734** of file **stm32l0xx_nucleo.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 **168** of file **stm32l0xx_nucleo.c**.

References **__STM32L0XX_NUCLEO_BSP_VERSION**.

void BSP_JOY_DeInit (void)

DeInit joystick GPIOs.

Note:
JOY DeInit does not disable the Mfx, just set the Mfx pins in Off mode

Return values:
None.

Definition at line **861** of file **stm32l0xx_nucleo.c**.

References **ADCx_DeInit()**.

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 **878** of file **stm32l0xx_nucleo.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 **841** of file **stm32l0xx_nucleo.c**.

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

void BSP_LED_DeInit (Led_TypeDef Led)

DeInit LEDs.

Parameters:

Led,: LED to be de-init. This parameter can be one of the following values:

- LED2

Note:

Led DeInit does not disable the GPIO clock nor disable the Mfx

Return values:

None

Definition at line **207** of file **stm32l0xx_nucleo.c**.

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

void BSP_LED_Init (Led_TypeDef Led)

Configures LED GPIO.

Parameters:

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

- LED2

Return values:

None

Definition at line **180** of file **stm32l0xx_nucleo.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:

- LED2

Return values:

None

Definition at line **237** of file **stm32l0xx_nucleo.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:

- LED2

Return values:

None

Definition at line **225** of file **stm32l0xx_nucleo.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:

- LED2

Return values:

None

Definition at line **249** of file **stm32l0xx_nucleo.c**.

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

void BSP_PB_DeInit (Button_TypeDef Button)

Push Button DeInit.

Parameters:

Button,: Button to be configured This parameter should be: BUTTON_USER

Note:
PB DeInit does not disable the GPIO clock

Return values:
None

Definition at line **303** of file **stm32l0xx_nucleo.c**.

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

uint32_t BSP_PB_GetState (Button_TypeDef Button)

Returns the selected Button state.

Parameters:
Button,: Specifies the Button to be checked. This parameter should be: BUTTON_USER

Return values:
Button state.

Definition at line **318** of file **stm32l0xx_nucleo.c**.

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

void BSP_PB_Init (Button_TypeDef Button,
ButtonMode_TypeDef ButtonMode
)

Configures Button GPIO and EXTI Line.

Parameters:

Button,: Specifies the Button to be configured. This parameter should be: BUTTON_USER

ButtonMode,: Specifies Button mode. This parameter can be one of following parameters:

- BUTTON_MODE_GPIO: Button will be used as simple IO
- BUTTON_MODE_EXTI: Button will be connected to EXTI line with interrupt generation capability

Return values:

None

Definition at line **265** of file **stm32l0xx_nucleo.c**.

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

void LCD_Delay (uint32_t Delay)

Wait for loop in ms.

Parameters:

Delay in ms.

Return values:

None

Definition at line **722** of file **stm32l0xx_nucleo.c**.

void LCD_IO_Init (void)

Initialize the LCD.

Return values:
None

Definition at line 593 of file stm32l0xx_nucleo.c.

References LCD_CS_GPIO_CLK_ENABLE, LCD_CS_HIGH, LCD_CS_PIN, LCD_DC_GPIO_CLK_ENABLE, LCD_DC_GPIO_PORT, LCD_DC_PIN, SD_CS_GPIO_PORT, 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 645 of file stm32l0xx_nucleo.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 **666** of file **stm32l0xx_nucleo.c**.

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

void LCD_IO_WriteReg (uint8_t LCDReg)

Write command to select the LCD register.

Parameters:

LCDReg,: Address of the selected register.

Return values:

None

Definition at line **624** of file **stm32l0xx_nucleo.c**.

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

void SD_IO_CSState (uint8_t val)

Set the SD_CS pin.

Parameters:

val,: pin value.

Return values:

None

Definition at line **525** of file **stm32l0xx_nucleo.c**.

References **SD_CS_HIGH**, and **SD_CS_LOW**.

void SD_IO_Init (void)

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

Return values:

None

Definition at line **482** of file **stm32l0xx_nucleo.c**.

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

void SD_IO_ReadData (uint8_t * DataOut,
uint16_t DataLength
)

Write an amount of data on the SD.

Parameters:

DataOut,: byte to send.

DataLength,: number of bytes to write

Return values:

none

Definition at line **570** of file **stm32l0xx_nucleo.c**.

References **SD_IO_WriteReadData()**.

uint8_t SD_IO_WriteByte (uint8_t Data)

Write a byte on the SD.

Parameters:

Data,: byte to send.

Return values:

Data written

Definition at line **555** of file **stm32l0xx_nucleo.c**.

References **SPIx_WriteReadData()**.

Referenced by **SD_IO_Init()**.

void SD_IO_WriteData (const uint8_t * Data,
uint16_t DataLength
)

Write an amount of data on the SD.

Parameters:

Data,: byte to send.

DataLength,: number of bytes to write

Return values:

none

Definition at line **582** of file **stm32l0xx_nucleo.c**.

References **SPIx_WriteData()**.

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

Write byte(s) on the SD.

Parameters:

DataIn,:	Pointer to data buffer to write
DataOut,:	Pointer to data buffer for read data
DataLength,:	number of bytes to write

Return values:

None

Definition at line **544** of file **stm32l0xx_nucleo.c**.

References **SPIx_WriteReadData()**.

Referenced by **SD_IO_ReadData()**.

static void SPIx_Error (void) [static]

SPI error treatment function.

Return values:

None

Definition at line **463** of file **stm32l0xx_nucleo.c**.

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

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

static void SPIx_Init (void) [static]

Initialize SPI HAL.

Return values:
None

Definition at line 367 of file stm32l0xx_nucleo.c.

References hnucleo_Spi, NUCLEO_SPIx, and SPIx_MspInit().

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

static void SPIx_MspInit (void) [static]

Initialize SPI MSP.

Return values:
None

Definition at line 332 of file stm32l0xx_nucleo.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 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 **444** of file **stm32l0xx_nucleo.c**.

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

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

static void SPIx_WriteData (uint8_t * DataIn,
uint16_t DataLength
) [static]

SPI Write an amount of data to device.

Parameters:

DataIn,: value to be written

DataLength,: number of bytes to write

Return values:

None

Definition at line **425** of file **stm32l0xx_nucleo.c**.

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

Referenced by **SD_IO_WriteData()**.

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

SPI Write a byte to device.

Parameters:

DataIn,:	value to be written
DataOut,:	read value
DataLength,:	number of bytes to write

Return values:

None

Definition at line **405** of file **stm32l0xx_nucleo.c**.

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

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

STM32L0xx_Nucleo 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_DeInit** (**Led_TypeDef** Led) DeInit LEDs.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** ButtonMode) Configures Button GPIO and EXTI Line.
void	**BSP_PB_DeInit** (**Button_TypeDef** Button) Push Button DeInit.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected Button state.
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.
void	**BSP_JOY_DeInit** (void) DeInit joystick GPIOs.

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 **168** of file **stm32l0xx_nucleo.c**.

References **__STM32L0XX_NUCLEO_BSP_VERSION**.

void BSP_JOY_DeInit (void)

DeInit joystick GPIOs.

Note:

JOY DeInit does not disable the Mfx, just set the Mfx pins in Off mode

Return values:

None.

Definition at line **861** of file **stm32l0xx_nucleo.c**.

References **ADCx_DeInit()**.

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 **878** of file **stm32l0xx_nucleo.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 **841** of file **stm32l0xx_nucleo.c**.

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

void BSP_LED_DeInit (Led_TypeDef Led)

DeInit LEDs.

Parameters:

Led,: LED to be de-init. This parameter can be one of the following values:

- LED2

Note:
Led DeInit does not disable the GPIO clock nor disable the Mfx

Return values:
None

Definition at line **207** of file **stm32l0xx_nucleo.c**.

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

void BSP_LED_Init (Led_TypeDef Led)

Configures LED GPIO.

Parameters:
Led,: Led to be configured. This parameter can be one of the following values:
- LED2

Return values:
None

Definition at line **180** of file **stm32l0xx_nucleo.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:
- LED2

Return values:

None

Definition at line **237** of file **stm32l0xx_nucleo.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:
- LED2

Return values:

None

Definition at line **225** of file **stm32l0xx_nucleo.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:
- LED2

Return values:

None

Definition at line **249** of file **stm32l0xx_nucleo.c**.

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

void BSP_PB_DeInit (Button_TypeDef Button)

Push Button DeInit.

Parameters:

Button,: Button to be configured This parameter should be: BUTTON_USER

Note:

PB DeInit does not disable the GPIO clock

Return values:

None

Definition at line **303** of file **stm32l0xx_nucleo.c**.

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

uint32_t BSP_PB_GetState (Button_TypeDef Button)

Returns the selected Button state.

Parameters:

Button,: Specifies the Button to be checked. This parameter should be: BUTTON_USER

Return values:

Button state.

Definition at line **318** of file **stm32l0xx_nucleo.c**.

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

void BSP_PB_Init (**Button_TypeDef** **Button,**
ButtonMode_TypeDef **ButtonMode**
)

Configures Button GPIO and EXTI Line.

Parameters:

Button,: Specifies the Button to be configured. This parameter should be: BUTTON_USER

ButtonMode,: Specifies Button mode. This parameter can be one of following parameters:

- BUTTON_MODE_GPIO: Button will be used as simple IO
- BUTTON_MODE_EXTI: Button will be connected to EXTI line with interrupt generation capability

Return values:

None

Definition at line **265** of file **stm32l0xx_nucleo.c**.

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

STM32L0xx_Nucleo BSP User Manual

Private Variables

STM32L0XX_NUCLEO_LOW_LEVEL

Variables

GPIO_TypeDef *	**LED_PORT** [**LEDn**] = {**LED2_GPIO_PORT**}
const uint16_t	**LED_PIN** [**LEDn**] = {**LED2_PIN**}
GPIO_TypeDef *	**BUTTON_PORT** [**BUTTONn**] = {**USER_BUTTON_GPIO_PORT** }
const uint16_t	**BUTTON_PIN** [**BUTTONn**] = {**USER_BUTTON_PIN** }
const uint8_t	**BUTTON_IRQn** [**BUTTONn**] = {**USER_BUTTON_EXTI_IRQn** }
uint32_t	**SpixTimeout** = **NUCLEO_SPIx_TIMEOUT_MAX** BUS variables.
static SPI_HandleTypeDef	**hnucleo_Spi**
static ADC_HandleTypeDef	**hnucleo_Adc**
static ADC_ChannelConfTypeDef	**sConfig**

Variable Documentation

const uint8_t BUTTON_IRQn[BUTTONn] = {USER_BUTTON_EXTI_I

Definition at line 101 of file stm32l0xx_nucleo.c.

Referenced by BSP_PB_DeInit(), and BSP_PB_Init().

const uint16_t BUTTON_PIN[BUTTONn] = {USER_BUTTON_PIN }

Definition at line 100 of file stm32l0xx_nucleo.c.

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

GPIO_TypeDef* BUTTON_PORT[BUTTONn] = {USER_BUTTON_GPI

Definition at line 99 of file stm32l0xx_nucleo.c.

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

ADC_HandleTypeDef hnucleo_Adc [static]

Definition at line 113 of file stm32l0xx_nucleo.c.

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

SPI_HandleTypeDef hnucleo_Spi [static]

Definition at line 109 of file stm32l0xx_nucleo.c.

Referenced by LCD_IO_WriteMultipleData(), SPIx_Error(), SPIx_Init(), SPIx_Write(), SPIx_WriteData(), and SPIx_WriteReadData().

const uint16_t LED_PIN[LEDn] = {LED2_PIN}

Definition at line 97 of file stm32l0xx_nucleo.c.

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

GPIO_TypeDef* LED_PORT[LEDn] = {LED2_GPIO_PORT}

Definition at line 95 of file stm32l0xx_nucleo.c.

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

ADC_ChannelConfTypeDef sConfig `[static]`

Definition at line 115 of file stm32l0xx_nucleo.c.

Referenced by BSP_JOY_Init().

uint32_t SpixTimeout = NUCLEO_SPIx_TIMEOUT_MAX

BUS variables.

Definition at line 108 of file stm32l0xx_nucleo.c.

Referenced by SPIx_Write(), SPIx_WriteData(), and SPIx_WriteReadData().

Generated on Mon Aug 28 2017 14:45:03 for STM32L0xx_Nucleo BSP
User Manual by  1.7.6.1

STM32L0xx_Nucleo BSP User Manual

Exported Types

STM32L0XX_NUCLEO_LOW_LEVEL

Enumerations

enum	**Led_TypeDef** { **LED2** = 0, **LED_GREEN** = LED2 }
enum	**Button_TypeDef** { **BUTTON_USER** = 0, **BUTTON_KEY** = BUTTON_USER }
enum	**ButtonMode_TypeDef** { **BUTTON_MODE_GPIO** = 0, **BUTTON_MODE_EXTI** = 1 }

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 Button_TypeDef

Enumerator:

BUTTON_USER

BUTTON_KEY

Definition at line 72 of file stm32l0xx_nucleo.h.

enum ButtonMode_TypeDef

Enumerator:

BUTTON_MODE_GPIO

BUTTON_MODE_EXTI

Definition at line 79 of file stm32l0xx_nucleo.h.

enum JOYState_TypeDef

Enumerator:

JOY_NONE

JOY_SEL

JOY_DOWN

JOY_LEFT

JOY_RIGHT

JOY_UP

Definition at line 85 of file stm32l0xx_nucleo.h.

enum Led_TypeDef

Enumerator:

LED2

LED_GREEN

Definition at line **65** of file **stm32l0xx_nucleo.h**.

STM32L0xx_Nucleo BSP User Manual

STM32L0XX_NUCLEO_LOW_LEVEL_BUTTON

Exported Constants

Defines

#define	**BUTTONn** 1
#define	**USER_BUTTON_PIN** GPIO_PIN_13 User push-button.
#define	**USER_BUTTON_GPIO_PORT** GPIOC
#define	**USER_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_GPIC
#define	**USER_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_GPI
#define	**USER_BUTTON_EXTI_IRQn** EXTI4_15_IRQn
#define	**KEY_BUTTON_PIN** **USER_BUTTON_PIN**
#define	**KEY_BUTTON_GPIO_PORT** **USER_BUTTON_GPIO_PORT**
#define	**KEY_BUTTON_GPIO_CLK_ENABLE**() **USER_BUTTON_GP**
#define	**KEY_BUTTON_GPIO_CLK_DISABLE**() **USER_BUTTON_G**
#define	**KEY_BUTTON_EXTI_IRQn** **USER_BUTTON_EXTI_IRQn**
#define	**BUTTONx_GPIO_CLK_ENABLE**(__INDEX__) do { if((__IND **USER_BUTTON_GPIO_CLK_ENABLE**();} while(0)
#define	**BUTTONx_GPIO_CLK_DISABLE**(__INDEX__) (((__INDEX_ **USER_BUTTON_GPIO_CLK_DISABLE**() : 0)

Define Documentation

#define BUTTONn 1

Definition at line 129 of file stm32l0xx_nucleo.h.

#define BUTTONx_GPIO_CLK_DISABLE (__INDEX__) (((__INDE

Definition at line 147 of file stm32l0xx_nucleo.h.

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

Definition at line 146 of file stm32l0xx_nucleo.h.

Referenced by BSP_PB_Init().

#define KEY_BUTTON_EXTI_IRQn USER_BUTTON_EXTI_IRQn

Definition at line 144 of file stm32l0xx_nucleo.h.

#define KEY_BUTTON_GPIO_CLK_DISABLE ( ) USER_BUTTON_

Definition at line 143 of file stm32l0xx_nucleo.h.

#define KEY_BUTTON_GPIO_CLK_ENABLE ( ) USER_BUTTON_C

Definition at line 142 of file stm32l0xx_nucleo.h.

#define KEY_BUTTON_GPIO_PORT USER_BUTTON_GPIO_PORT

Definition at line **141** of file **stm32l0xx_nucleo.h**.

#define KEY_BUTTON_PIN USER_BUTTON_PIN

Definition at line **140** of file **stm32l0xx_nucleo.h**.

#define USER_BUTTON_EXTI_IRQn EXTI4_15_IRQn

Definition at line **138** of file **stm32l0xx_nucleo.h**.

#define USER_BUTTON_GPIO_CLK_DISABLE ( ) __HAL_RCC_GI

Definition at line **137** of file **stm32l0xx_nucleo.h**.

#define USER_BUTTON_GPIO_CLK_ENABLE ( ) __HAL_RCC_GP

Definition at line **136** of file **stm32l0xx_nucleo.h**.

#define USER_BUTTON_GPIO_PORT GPIOC

Definition at line **135** of file **stm32l0xx_nucleo.h**.

#define USER_BUTTON_PIN GPIO_PIN_13

User push-button.

Definition at line **134** of file **stm32l0xx_nucleo.h**.

STM32L0xx_Nucleo BSP User Manual

STM32L0XX_NUCLEO_LOW_LEVEL_COMPONE

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 (shield D4)
#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 (shield D10)
#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_CLK
#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_CLK
#define	**NUCLEO_ADCx** ADC1 ADC Interface pins used to detect motion of Joystick available

shield.

```
#define NUCLEO_ADCx_CLK_ENABLE()   __HAL_RCC_ADC1_CLK
#define NUCLEO_ADCx_CLK_DISABLE()  __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_GPIO
#define NUCLEO_ADCx_GPIO_CLK_DISABLE()  __HAL_RCC_GPI
```

Define Documentation

#define LCD_CS_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOB_C

Definition at line 215 of file stm32l0xx_nucleo.h.

#define LCD_CS_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOB_CL

Definition at line 214 of file stm32l0xx_nucleo.h.

Referenced by LCD_IO_Init().

#define LCD_CS_GPIO_PORT GPIOB

Definition at line 213 of file stm32l0xx_nucleo.h.

#define LCD_CS_HIGH ( ) HAL_GPIO_WritePin(LCD_CS_GPIO_P

Definition at line 197 of file stm32l0xx_nucleo.h.

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

#define LCD_CS_LOW ( ) HAL_GPIO_WritePin(LCD_CS_GPIO_P

LCD Control Lines management.

Definition at line 196 of file stm32l0xx_nucleo.h.

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

#define LCD_CS_PIN GPIO_PIN_6

LCD Control Interface pins (shield D10)

Definition at line 212 of file stm32l0xx_nucleo.h.

Referenced by LCD_IO_Init(), and SD_IO_Init().

#define LCD_DC_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOA_C

Definition at line 223 of file stm32l0xx_nucleo.h.

#define LCD_DC_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOA_CL

Definition at line 222 of file stm32l0xx_nucleo.h.

Referenced by LCD_IO_Init().

#define LCD_DC_GPIO_PORT GPIOA

Definition at line 221 of file stm32l0xx_nucleo.h.

Referenced by LCD_IO_Init().

#define LCD_DC_HIGH ( ) HAL_GPIO_WritePin(LCD_DC_GPIO_P

Definition at line 199 of file stm32l0xx_nucleo.h.

Referenced by LCD_IO_WriteData(), and LCD_IO_WriteMultipleData().

#define LCD_DC_LOW ( ) HAL_GPIO_WritePin(LCD_DC_GPIO_P(

Definition at line 198 of file stm32l0xx_nucleo.h.

Referenced by LCD_IO_WriteReg().

#define LCD_DC_PIN GPIO_PIN_9

LCD Data/Command Interface pins.

Definition at line 220 of file stm32l0xx_nucleo.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 231 of file stm32l0xx_nucleo.h.

Referenced by ADCx_DeInit(), and ADCx_Init().

#define NUCLEO_ADCx_CLK_DISABLE ( ) __HAL_RCC_ADC1_C

Definition at line 233 of file stm32l0xx_nucleo.h.

Referenced by ADCx_MspDeInit().

#define NUCLEO_ADCx_CLK_ENABLE ( ) __HAL_RCC_ADC1_CL

Definition at line 232 of file stm32l0xx_nucleo.h.

Referenced by ADCx_MspInit().

#define NUCLEO_ADCx_GPIO_CLK_DISABLE ( ) __HAL_RCC_GI

Definition at line 238 of file stm32l0xx_nucleo.h.

#define NUCLEO_ADCx_GPIO_CLK_ENABLE ( ) __HAL_RCC_GP

Definition at line 237 of file stm32l0xx_nucleo.h.

Referenced by ADCx_MspInit().

#define NUCLEO_ADCx_GPIO_PIN GPIO_PIN_0

Definition at line 236 of file stm32l0xx_nucleo.h.

Referenced by ADCx_MspDeInit(), and ADCx_MspInit().

#define NUCLEO_ADCx_GPIO_PORT GPIOB

Definition at line 235 of file stm32l0xx_nucleo.h.

Referenced by ADCx_MspDeInit(), and ADCx_MspInit().

#define SD_CS_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOB_CL

Definition at line 207 of file stm32l0xx_nucleo.h.

#define SD_CS_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOB_CLK

Definition at line 206 of file stm32l0xx_nucleo.h.

Referenced by **SD_IO_Init()**.

#define SD_CS_GPIO_PORT GPIOB

Definition at line **205** of file **stm32l0xx_nucleo.h**.

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

#define SD_CS_HIGH ( ) HAL_GPIO_WritePin(SD_CS_GPIO_POR

Definition at line **191** of file **stm32l0xx_nucleo.h**.

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

#define SD_CS_LOW ( ) HAL_GPIO_WritePin(SD_CS_GPIO_POR

SD Control Lines management.

Definition at line **190** of file **stm32l0xx_nucleo.h**.

Referenced by **SD_IO_CSState()**.

#define SD_CS_PIN GPIO_PIN_5

SD Control Interface pins (shield D4)

Definition at line **204** of file **stm32l0xx_nucleo.h**.

Referenced by **SD_IO_Init()**.

STM32L0xx_Nucleo BSP User Manual

STM32L0XX_NUCLEO_LOW_LEVEL_LED

Exported Constants

Define for STM32L0XX_NUCLEO board. More...

Defines

#define	**LEDn** 1
#define	**LED2_PIN** GPIO_PIN_5
#define	**LED2_GPIO_PORT** GPIOA
#define	**LED2_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOA_CLK_E
#define	**LED2_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOA_CLK_D
#define	**LEDx_GPIO_CLK_ENABLE**(__INDEX__) do { if((__INDEX_ **LED2_GPIO_CLK_ENABLE**();} while(0)
#define	**LEDx_GPIO_CLK_DISABLE**(__INDEX__) (((__INDEX__) == **LED2_GPIO_CLK_DISABLE**() : 0)

Detailed Description

Define for STM32L0XX_NUCLEO board.

Define Documentation

#define LED2_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOA_CLK

Definition at line 118 of file stm32l0xx_nucleo.h.

#define LED2_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOA_CLK_

Definition at line 117 of file stm32l0xx_nucleo.h.

#define LED2_GPIO_PORT GPIOA

Definition at line 116 of file stm32l0xx_nucleo.h.

#define LED2_PIN GPIO_PIN_5

Definition at line 115 of file stm32l0xx_nucleo.h.

#define LEDn 1

Definition at line 113 of file stm32l0xx_nucleo.h.

#define LEDx_GPIO_CLK_DISABLE (__INDEX__) (((__INDEX__)

Definition at line 121 of file stm32l0xx_nucleo.h.

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

Definition at line 120 of file stm32l0xx_nucleo.h.

Referenced by **BSP_LED_Init()**.

STM32L0xx_Nucleo BSP User Manual

STM32L0XX_NUCLEO_LOW_LEVEL_BUS

Exported Constants

Defines

#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_5
#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** GPIOA
#define	**NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_ENABLE**() __HAI
#define	**NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_DISABLE**() __HA
#define	**NUCLEO_SPIx_MISO_PIN** GPIO_PIN_6
#define	**NUCLEO_SPIx_MOSI_PIN** GPIO_PIN_7
#define	**NUCLEO_SPIx_TIMEOUT_MAX** 1000

Define Documentation

#define NUCLEO_SPIx SPI1

Definition at line 157 of file stm32l0xx_nucleo.h.

Referenced by SPIx_Init().

#define NUCLEO_SPIx_CLK_ENABLE ( ) __HAL_RCC_SPI1_CLK

Definition at line 158 of file stm32l0xx_nucleo.h.

Referenced by SPIx_MspInit().

#define NUCLEO_SPIx_MISO_MOSI_AF GPIO_AF0_SPI1

Definition at line 166 of file stm32l0xx_nucleo.h.

Referenced by SPIx_MspInit().

#define NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_DISABLE ( ) __H

Definition at line 169 of file stm32l0xx_nucleo.h.

#define NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_ENABLE ( ) __H

Definition at line 168 of file stm32l0xx_nucleo.h.

Referenced by SPIx_MspInit().

#define NUCLEO_SPIx_MISO_MOSI_GPIO_PORT GPIOA

Definition at line **167** of file **stm32l0xx_nucleo.h**.

Referenced by **SPIx_MspInit()**.

#define NUCLEO_SPIx_MISO_PIN GPIO_PIN_6

Definition at line **170** of file **stm32l0xx_nucleo.h**.

Referenced by **SPIx_MspInit()**.

#define NUCLEO_SPIx_MOSI_PIN GPIO_PIN_7

Definition at line **171** of file **stm32l0xx_nucleo.h**.

Referenced by **SPIx_MspInit()**.

#define NUCLEO_SPIx_SCK_AF GPIO_AF0_SPI1

Definition at line **160** of file **stm32l0xx_nucleo.h**.

Referenced by **SPIx_MspInit()**.

#define NUCLEO_SPIx_SCK_GPIO_CLK_DISABLE ( ) __HAL_RC

Definition at line **164** of file **stm32l0xx_nucleo.h**.

#define NUCLEO_SPIx_SCK_GPIO_CLK_ENABLE ( ) __HAL_RCC

Definition at line **163** of file **stm32l0xx_nucleo.h**.

Referenced by **SPIx_MspInit()**.

#define NUCLEO_SPIx_SCK_GPIO_PORT GPIOA

Definition at line 161 of file stm32l0xx_nucleo.h.

Referenced by SPIx_MspInit().

#define NUCLEO_SPIx_SCK_PIN GPIO_PIN_5

Definition at line 162 of file stm32l0xx_nucleo.h.

Referenced by SPIx_MspInit().

#define NUCLEO_SPIx_TIMEOUT_MAX 1000

Definition at line 177 of file stm32l0xx_nucleo.h.

STM32L0xx_Nucleo BSP User Manual

Private Function Prototypes

STM32L0XX_NUCLEO_LOW_LEVEL

Functions

uint8_t **SD_IO_ReadByte** (void)

Function Documentation

uint8_t SD_IO_ReadByte (void)

STM32L0xx_Nucleo BSP User Manual

Firmware Directory Reference

Directories

directory	**Drivers**

STM32L0xx_Nucleo BSP User Manual

Drivers Directory Reference

Directories

directory	**BSP**

STM32L0xx_Nucleo BSP User Manual

BSP Directory Reference

Directories

directory	**STM32L0xx_Nucleo**

STM32L0xx_Nucleo BSP User Manual

STM32L0xx_Nucleo Directory Reference

Files

file	**stm32l0xx_nucleo.c** [code] This file provides set of firmware functions to manage:
file	**stm32l0xx_nucleo.h** [code] This file contains definitions for:

STM32L0xx_Nucleo BSP User Manual

stm32l0xx_nucleo.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l0xx_nucleo.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_H
00042 #define __STM32L0XX_NUCLEO_H
00043
00044 #ifdef __cplusplus
00045  extern "C" {
00046 #endif
00047
00048 /** @addtogroup BSP
00049   * @{
00050   */
00051
00052 /** @addtogroup STM32L0XX_NUCLEO
00053   * @{
00054   */
00055
00056 /* Includes ----------------------------------
-----------------------------------*/
00057 #include "stm32l0xx_hal.h"
00058 /** @addtogroup STM32L0XX_NUCLEO_LOW_LEVEL
00059   * @{
00060   */
00061
00062 /** @defgroup STM32L0XX_NUCLEO_LOW_LEVEL_Exp
orted_Types Exported Types
00063   * @{
00064   */
00065 typedef enum
00066 {
00067   LED2 = 0,
00068
```

```
00069   LED_GREEN = LED2
00070 } Led_TypeDef;
00071
00072 typedef enum
00073 {
00074   BUTTON_USER = 0,
00075   /* Alias */
00076   BUTTON_KEY  = BUTTON_USER
00077 } Button_TypeDef;
00078
00079 typedef enum
00080 {
00081   BUTTON_MODE_GPIO = 0,
00082   BUTTON_MODE_EXTI = 1
00083 } ButtonMode_TypeDef;
00084
00085 typedef enum
00086 {
00087   JOY_NONE  = 0,
00088   JOY_SEL   = 1,
00089   JOY_DOWN  = 2,
00090   JOY_LEFT  = 3,
00091   JOY_RIGHT = 4,
00092   JOY_UP    = 5
00093 } JOYState_TypeDef;
00094
00095 /**
00096   * @}
00097   */
00098
00099 /** @defgroup STM32L0XX_NUCLEO_LOW_LEVEL_Exp
orted_Constants Exported Constants
00100   * @{
00101   */
00102
00103 /**
00104 * @brief       Define for STM32L0XX_NUCLEO
```

```
board
00105   */
00106 #if !defined (USE_STM32L0XX_NUCLEO)
00107  #define USE_STM32L0XX_NUCLEO
00108 #endif
00109
00110 /** @addtogroup STM32L0XX_NUCLEO_LOW_LEVEL_L
ED
00111   * @{
00112   */
00113 #define LEDn                                    1
00114
00115 #define LED2_PIN                                G
PIO_PIN_5
00116 #define LED2_GPIO_PORT                          G
PIOA
00117 #define LED2_GPIO_CLK_ENABLE()                  __H
AL_RCC_GPIOA_CLK_ENABLE()
00118 #define LED2_GPIO_CLK_DISABLE()                 __H
AL_RCC_GPIOA_CLK_DISABLE()
00119
00120 #define LEDx_GPIO_CLK_ENABLE(__INDEX__)   do
 { if((__INDEX__) == 0) LED2_GPIO_CLK_ENABLE();} w
hile(0)
00121 #define LEDx_GPIO_CLK_DISABLE(__INDEX__)  ((
(__INDEX__) == 0) ? LED2_GPIO_CLK_DISABLE() : 0)
00122 /**
00123   * @}
00124   */
00125
00126 /** @addtogroup STM32L0XX_NUCLEO_LOW_LEVEL_B
UTTON
00127   * @{
00128   */
00129 #define BUTTONn                                 1
00130
00131 /**
```

```
00132   * @brief User push-button
00133   */
00134 #define USER_BUTTON_PIN
    GPIO_PIN_13
00135 #define USER_BUTTON_GPIO_PORT
    GPIOC
00136 #define USER_BUTTON_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOC_CLK_ENABLE()
00137 #define USER_BUTTON_GPIO_CLK_DISABLE()
    __HAL_RCC_GPIOC_CLK_DISABLE()
00138 #define USER_BUTTON_EXTI_IRQn
    EXTI4_15_IRQn
00139 /* Aliases */
00140 #define KEY_BUTTON_PIN
    USER_BUTTON_PIN
00141 #define KEY_BUTTON_GPIO_PORT
    USER_BUTTON_GPIO_PORT
00142 #define KEY_BUTTON_GPIO_CLK_ENABLE()
    USER_BUTTON_GPIO_CLK_ENABLE()
00143 #define KEY_BUTTON_GPIO_CLK_DISABLE()
    USER_BUTTON_GPIO_CLK_DISABLE()
00144 #define KEY_BUTTON_EXTI_IRQn
    USER_BUTTON_EXTI_IRQn
00145
00146 #define BUTTONx_GPIO_CLK_ENABLE(__INDEX__)
  do { if((__INDEX__) == 0) USER_BUTTON_GPIO_CLK_E
NABLE();} while(0)
00147 #define BUTTONx_GPIO_CLK_DISABLE(__INDEX__)
  (((__INDEX__) == 0) ? USER_BUTTON_GPIO_CLK_DISAB
LE() : 0)
00148 /**
00149   * @}
00150   */
00151
00152 /** @addtogroup STM32L0XX_NUCLEO_LOW_LEVEL_B
US
00153   * @{
```

```
00154   */
00155 #if defined(HAL_SPI_MODULE_ENABLED)
00156 /*########################## SPI1 ##############
#######################*/
00157 #define NUCLEO_SPIx
        SPI1
00158 #define NUCLEO_SPIx_CLK_ENABLE()
      __HAL_RCC_SPI1_CLK_ENABLE()
00159
00160 #define NUCLEO_SPIx_SCK_AF
        GPIO_AF0_SPI1
00161 #define NUCLEO_SPIx_SCK_GPIO_PORT
        GPIOA
00162 #define NUCLEO_SPIx_SCK_PIN
        GPIO_PIN_5
00163 #define NUCLEO_SPIx_SCK_GPIO_CLK_ENABLE()
      __HAL_RCC_GPIOA_CLK_ENABLE()
00164 #define NUCLEO_SPIx_SCK_GPIO_CLK_DISABLE()
      __HAL_RCC_GPIOA_CLK_DISABLE()
00165
00166 #define NUCLEO_SPIx_MISO_MOSI_AF
        GPIO_AF0_SPI1
00167 #define NUCLEO_SPIx_MISO_MOSI_GPIO_PORT
        GPIOA
00168 #define NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_ENABL
E()   __HAL_RCC_GPIOA_CLK_ENABLE()
00169 #define NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_DISAB
LE()  __HAL_RCC_GPIOA_CLK_DISABLE()
00170 #define NUCLEO_SPIx_MISO_PIN
        GPIO_PIN_6
00171 #define NUCLEO_SPIx_MOSI_PIN
        GPIO_PIN_7
00172 /* Maximum Timeout values for flags waiting
loops. These timeouts are not based
00173    on accurate values, they just guarantee t
hat the application will not remain
00174    stuck if the SPI communication is corrupt
```

```
ed.
00175    You may modify these timeout values depen
ding on CPU frequency and application
00176    conditions (interrupts routines ...). */

00177 #define NUCLEO_SPIx_TIMEOUT_MAX
      1000
00178 #endif /* HAL_SPI_MODULE_ENABLED */
00179 /**
00180   * @}
00181   */
00182
00183 /** @addtogroup STM32L0XX_NUCLEO_LOW_LEVEL_C
OMPONENT
00184   * @{
00185   */
00186
00187 /**
00188   * @brief  SD Control Lines management
00189   */
00190 #define SD_CS_LOW()       HAL_GPIO_WritePin(
SD_CS_GPIO_PORT, SD_CS_PIN, GPIO_PIN_RESET)
00191 #define SD_CS_HIGH()      HAL_GPIO_WritePin(
SD_CS_GPIO_PORT, SD_CS_PIN, GPIO_PIN_SET)
00192
00193 /**
00194   * @brief  LCD Control Lines management
00195   */
00196 #define LCD_CS_LOW()      HAL_GPIO_WritePin(
LCD_CS_GPIO_PORT, LCD_CS_PIN, GPIO_PIN_RESET)
00197 #define LCD_CS_HIGH()     HAL_GPIO_WritePin(
LCD_CS_GPIO_PORT, LCD_CS_PIN, GPIO_PIN_SET)
00198 #define LCD_DC_LOW()      HAL_GPIO_WritePin(
LCD_DC_GPIO_PORT, LCD_DC_PIN, GPIO_PIN_RESET)
00199 #define LCD_DC_HIGH()     HAL_GPIO_WritePin(
LCD_DC_GPIO_PORT, LCD_DC_PIN, GPIO_PIN_SET)
00200
```

```
00201 /**
00202   * @brief  SD Control Interface pins (shiel
d D4)
00203   */
00204 #define SD_CS_PIN
      GPIO_PIN_5
00205 #define SD_CS_GPIO_PORT
      GPIOB
00206 #define SD_CS_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOB_CLK_ENABLE()
00207 #define SD_CS_GPIO_CLK_DISABLE()
    __HAL_RCC_GPIOB_CLK_DISABLE()
00208
00209 /**
00210   * @brief  LCD Control Interface pins (shie
ld D10)
00211   */
00212 #define LCD_CS_PIN
       GPIO_PIN_6
00213 #define LCD_CS_GPIO_PORT
       GPIOB
00214 #define LCD_CS_GPIO_CLK_ENABLE()
     __HAL_RCC_GPIOB_CLK_ENABLE()
00215 #define LCD_CS_GPIO_CLK_DISABLE()
     __HAL_RCC_GPIOB_CLK_DISABLE()
00216
00217 /**
00218   * @brief  LCD Data/Command Interface pins
00219   */
00220 #define LCD_DC_PIN
       GPIO_PIN_9
00221 #define LCD_DC_GPIO_PORT
       GPIOA
00222 #define LCD_DC_GPIO_CLK_ENABLE()
     __HAL_RCC_GPIOA_CLK_ENABLE()
00223 #define LCD_DC_GPIO_CLK_DISABLE()
     __HAL_RCC_GPIOA_CLK_DISABLE()
```

```
00224
00225 #if defined(HAL_ADC_MODULE_ENABLED)
00226 /*##################### ADC1 ###############
#####################*/
00227 /**
00228   * @brief  ADC Interface pins
00229   *         used to detect motion of Joystic
k available on Adafruit 1.8 TFT shield
00230   */
00231 #define NUCLEO_ADCx
        ADC1
00232 #define NUCLEO_ADCx_CLK_ENABLE()
        __HAL_RCC_ADC1_CLK_ENABLE()
00233 #define NUCLEO_ADCx_CLK_DISABLE()
        __HAL_RCC_ADC1_CLK_DISABLE()
00234
00235 #define NUCLEO_ADCx_GPIO_PORT
        GPIOB
00236 #define NUCLEO_ADCx_GPIO_PIN
        GPIO_PIN_0
00237 #define NUCLEO_ADCx_GPIO_CLK_ENABLE()
        __HAL_RCC_GPIOB_CLK_ENABLE()
00238 #define NUCLEO_ADCx_GPIO_CLK_DISABLE()
        __HAL_RCC_GPIOB_CLK_DISABLE()
00239
00240 #endif /* HAL_ADC_MODULE_ENABLED */
00241
00242 /**
00243   * @}
00244   */
00245
00246 /** @defgroup STM32L0XX_NUCLEO_LOW_LEVEL_Exp
orted_Macros Exported Macros
00247   * @{
00248   */
00249 /**
00250   * @}
```

```
00251   */
00252
00253
00254 /** @defgroup STM32L0XX_NUCLEO_LOW_LEVEL_Exp
orted_Functions Exported Functions
00255   * @{
00256   */
00257 uint32_t         BSP_GetVersion(void);
00258 void             BSP_LED_Init(Led_TypeDef Le
d);
00259 void             BSP_LED_DeInit(Led_TypeDef
Led);
00260 void             BSP_LED_On(Led_TypeDef Led)
;
00261 void             BSP_LED_Off(Led_TypeDef Led
);
00262 void             BSP_LED_Toggle(Led_TypeDef
Led);
00263 void             BSP_PB_Init(Button_TypeDef
Button, ButtonMode_TypeDef ButtonMode);
00264 void             BSP_PB_DeInit(Button_TypeDef
 Button);
00265 uint32_t         BSP_PB_GetState(Button_Type
Def Button);
00266 #if defined(HAL_ADC_MODULE_ENABLED)
00267 uint8_t          BSP_JOY_Init(void);
00268 JOYState_TypeDef BSP_JOY_GetState(void);
00269 void             BSP_JOY_DeInit(void);
00270 #endif /* HAL_ADC_MODULE_ENABLED */
00271 /**
00272   * @}
00273   */
00274
00275 /**
00276   * @}
00277   */
00278
```

```
00279 /**
00280   * @}
00281   */
00282
00283 /**
00284   * @}
00285   */
00286
00287 /**
00288   * @}
00289   */
00290
00291 #ifdef __cplusplus
00292 }
00293 #endif
00294
00295 #endif /* __STM32L0XX_NUCLEO_H */
00296
00297 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32L0xx_Nucleo BSP User Manual

stm32l0xx_nucleo.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l0xx_nucleo.c
00004   * @author  MCD Application Team
00005   * @brief   This file provides set of firmware functions to manage:
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 /* Includes ------------------------------------------------------------------*/
00041 #include "stm32l0xx_nucleo.h"
00042
00043 /** @addtogroup BSP
00044   * @{
00045   */
00046
00047 /** @addtogroup STM32L0XX_NUCLEO
00048   * @{
00049   */
00050
00051 /** @addtogroup STM32L0XX_NUCLEO_LOW_LEVEL
00052   * @brief This file provides set of firmware functions to manage Leds and push-button
00053   *        available on STM32L0XX-Nucleo Kit from STMicroelectronics.
00054   * @{
00055   */
00056
00057 /** @defgroup STM32L0XX_NUCLEO_LOW_LEVEL_Private_TypesDefinitions Private Types Definitions
00058   * @{
00059   */
00060 /**
00061   * @}
00062   */
00063
00064
00065 /** @defgroup STM32L0XX_NUCLEO_LOW_LEVEL_Private_Defines Private Defines
00066   * @{
```

```
00067   */
00068
00069 /**
00070   * @brief STM32L0XX NUCLEO BSP Driver versi
on number
00071   */
00072 #define __STM32L0XX_NUCLEO_BSP_VERSION_MAIN
  (0x02) /*!< [31:24] main version */
00073 #define __STM32L0XX_NUCLEO_BSP_VERSION_SUB1
  (0x01) /*!< [23:16] sub1 version */
00074 #define __STM32L0XX_NUCLEO_BSP_VERSION_SUB2
  (0x01) /*!< [15:8]  sub2 version */
00075 #define __STM32L0XX_NUCLEO_BSP_VERSION_RC
  (0x00) /*!< [7:0]  release candidate */
00076 #define __STM32L0XX_NUCLEO_BSP_VERSION
   ((__STM32L0XX_NUCLEO_BSP_VERSION_MAIN << 24)\
00077
 |(__STM32L0XX_NUCLEO_BSP_VERSION_SUB1 << 16)\
00078
 |(__STM32L0XX_NUCLEO_BSP_VERSION_SUB2 << 8 )\
00079
 |(__STM32L0XX_NUCLEO_BSP_VERSION_RC))
00080
00081 /**
00082   * @brief LINK SD Card
00083   */
00084 #define SD_DUMMY_BYTE            0xFF
00085 #define SD_NO_RESPONSE_EXPECTED  0x80
00086
00087 /**
00088   * @}
00089   */
00090
00091
00092 /** @defgroup STM32L0XX_NUCLEO_LOW_LEVEL_Pri
vate_Variables Private Variables
00093   * @{
```

```
00094   */
00095 GPIO_TypeDef* LED_PORT[LEDn] = {LED2_GPIO_PO
RT};
00096
00097 const uint16_t LED_PIN[LEDn] = {LED2_PIN};
00098
00099 GPIO_TypeDef* BUTTON_PORT[BUTTONn] = {USER_B
UTTON_GPIO_PORT };
00100 const uint16_t BUTTON_PIN[BUTTONn] = {USER_B
UTTON_PIN };
00101 const uint8_t BUTTON_IRQn[BUTTONn] = {USER_B
UTTON_EXTI_IRQn };
00102
00103 /**
00104  * @brief BUS variables
00105  */
00106
00107 #ifdef HAL_SPI_MODULE_ENABLED
00108 uint32_t SpixTimeout = NUCLEO_SPIx_TIMEOUT_M
AX; /*<! Value of Timeout when SPI communication f
ails */
00109 static SPI_HandleTypeDef hnucleo_Spi;
00110 #endif /* HAL_SPI_MODULE_ENABLED */
00111
00112 #ifdef HAL_ADC_MODULE_ENABLED
00113 static ADC_HandleTypeDef hnucleo_Adc;
00114 /* ADC channel configuration structure decla
ration */
00115 static ADC_ChannelConfTypeDef sConfig;
00116 #endif /* HAL_ADC_MODULE_ENABLED */
00117
00118 /**
00119   * @}
00120   */
00121
00122 /** @defgroup STM32L0XX_NUCLEO_LOW_LEVEL_Pri
vate_FunctionPrototypes Private Function Prototypes
```

```
00123   * @{
00124   */
00125 #ifdef HAL_SPI_MODULE_ENABLED
00126 static void                SPIx_Init(void);
00127 static void                SPIx_Write(uint8_t
 Value);
00128 static void                SPIx_WriteData(uin
t8_t *DataIn, uint16_t DataLength);
00129 static void                SPIx_WriteReadData(
const uint8_t *DataIn, uint8_t *DataOut, uint16_t
DataLegnth);
00130 static void                SPIx_Error (void);
00131 static void                SPIx_MspInit(void)
;
00132
00133 /* SD IO functions */
00134 void                       SD_IO_Init(void);
00135 void                       SD_IO_CSState(uint
8_t state);
00136 void                       SD_IO_WriteReadData
(const uint8_t *DataIn, uint8_t *DataOut, uint16_t
 DataLength);
00137 void                       SD_IO_ReadData(uin
t8_t *DataOut, uint16_t DataLength);
00138 void                       SD_IO_WriteData(co
nst uint8_t *Data, uint16_t DataLength);
00139 uint8_t                    SD_IO_WriteByte(ui
nt8_t Data);
00140 uint8_t                    SD_IO_ReadByte(void
);
00141
00142 /* LCD IO functions */
00143 void                       LCD_IO_Init(void);
00144 void                       LCD_IO_WriteData(u
int8_t Data);
00145 void                       LCD_IO_WriteMultip
```

```
leData(uint8_t *pData, uint32_t Size);
00146 void                    LCD_IO_WriteReg(ui
nt8_t LCDReg);
00147 void                    LCD_Delay(uint32_t
 delay);
00148 #endif /* HAL_SPI_MODULE_ENABLED */
00149
00150 #ifdef HAL_ADC_MODULE_ENABLED
00151 static HAL_StatusTypeDef              ADCx_
Init(void);
00152 static void              ADCx_DeInit(void);
00153 static void              ADCx_MspInit(ADC_H
andleTypeDef *hadc);
00154 static void              ADCx_MspDeInit(ADC
_HandleTypeDef *hadc);
00155 #endif /* HAL_ADC_MODULE_ENABLED */
00156 /**
00157   * @}
00158   */
00159
00160 /** @defgroup STM32L0XX_NUCLEO_LOW_LEVEL_Pri
vate_Functions Private Functions
00161   * @{
00162   */
00163
00164 /**
00165   * @brief  This method returns the STM32L0X
X NUCLEO BSP Driver revision
00166   * @retval version : 0xXYZR (8bits for each
 decimal, R for RC)
00167   */
00168 uint32_t BSP_GetVersion(void)
00169 {
00170   return __STM32L0XX_NUCLEO_BSP_VERSION;
00171 }
00172
00173 /**
```

```
00174   * @brief  Configures LED GPIO.
00175   * @param  Led: Led to be configured.
00176   *          This parameter can be one of th
e following values:
00177   *            @arg  LED2
00178   * @retval None
00179   */
00180 void BSP_LED_Init(Led_TypeDef Led)
00181 {
00182   GPIO_InitTypeDef  gpioinitstruct;
00183
00184   /* Enable the GPIO_LED Clock */
00185   LEDx_GPIO_CLK_ENABLE(Led);
00186
00187   /* Configure the GPIO_LED pin */
00188   gpioinitstruct.Pin = LED_PIN[Led];
00189   gpioinitstruct.Mode = GPIO_MODE_OUTPUT_PP;
00190   gpioinitstruct.Pull = GPIO_NOPULL;
00191   gpioinitstruct.Speed = GPIO_SPEED_FREQ_VER
Y_HIGH;
00192
00193   HAL_GPIO_Init(LED_PORT[Led], &gpioinitstru
ct);
00194
00195   /* Reset PIN to switch off the LED */
00196   HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[L
ed], GPIO_PIN_RESET);
00197 }
00198
00199 /**
00200   * @brief  DeInit LEDs.
00201   * @param  Led: LED to be de-init.
00202   *   This parameter can be one of the follo
wing values:
00203   *     @arg  LED2
00204   * @note Led DeInit does not disable the GP
IO clock nor disable the Mfx
```

```
00205   * @retval None
00206   */
00207 void BSP_LED_DeInit(Led_TypeDef Led)
00208 {
00209   GPIO_InitTypeDef  gpio_init_structure;
00210
00211   /* Turn off LED */
00212   HAL_GPIO_WritePin(LED_PORT[Led],LED_PIN[Le
d], GPIO_PIN_RESET);
00213   /* DeInit the GPIO_LED pin */
00214   gpio_init_structure.Pin = LED_PIN[Led];
00215   HAL_GPIO_DeInit(LED_PORT[Led], gpio_init_s
tructure.Pin);
00216 }
00217
00218 /**
00219   * @brief  Turns selected LED On.
00220   * @param  Led: Specifies the Led to be set
 on.
00221   *   This parameter can be one of following
 parameters:
00222   *            @arg  LED2
00223   * @retval None
00224   */
00225 void BSP_LED_On(Led_TypeDef Led)
00226 {
00227   HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[L
ed], GPIO_PIN_SET);
00228 }
00229
00230 /**
00231   * @brief  Turns selected LED Off.
00232   * @param  Led: Specifies the Led to be set
 off.
00233   *   This parameter can be one of following
 parameters:
00234   *            @arg  LED2
```

```
00235   * @retval None
00236   */
00237 void BSP_LED_Off(Led_TypeDef Led)
00238 {
00239   HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[L
ed], GPIO_PIN_RESET);
00240 }
00241
00242 /**
00243   * @brief  Toggles the selected LED.
00244   * @param  Led: Specifies the Led to be tog
gled.
00245   *   This parameter can be one of following
 parameters:
00246   *            @arg  LED2
00247   * @retval None
00248   */
00249 void BSP_LED_Toggle(Led_TypeDef Led)
00250 {
00251   HAL_GPIO_TogglePin(LED_PORT[Led], LED_PIN[
Led]);
00252 }
00253
00254 /**
00255   * @brief  Configures Button GPIO and EXTI
Line.
00256   * @param  Button: Specifies the Button to
be configured.
00257   *   This parameter should be: BUTTON_USER
00258   * @param  ButtonMode: Specifies Button mod
e.
00259   *   This parameter can be one of following
 parameters:
00260   *     @arg  BUTTON_MODE_GPIO: Button will
be used as simple IO
00261   *     @arg BUTTON_MODE_EXTI: Button will b
e connected to EXTI line with interrupt
```

```
00262   *                                      generation ca
pability
00263   * @retval None
00264   */
00265 void BSP_PB_Init(Button_TypeDef Button, Butt
onMode_TypeDef ButtonMode)
00266 {
00267   GPIO_InitTypeDef gpioinitstruct;
00268
00269   /* Enable the BUTTON Clock */
00270   BUTTONx_GPIO_CLK_ENABLE(Button);
00271
00272   gpioinitstruct.Pin = BUTTON_PIN[Button];
00273   gpioinitstruct.Pull = GPIO_NOPULL;
00274   gpioinitstruct.Speed = GPIO_SPEED_FREQ_MED
IUM;
00275
00276   if(ButtonMode == BUTTON_MODE_GPIO)
00277   {
00278     /* Configure Button pin as input */
00279     gpioinitstruct.Mode = GPIO_MODE_INPUT;
00280
00281     HAL_GPIO_Init(BUTTON_PORT[Button], &gpio
initstruct);
00282   }
00283
00284   if(ButtonMode == BUTTON_MODE_EXTI)
00285   {
00286     /* Configure Button pin as input with Ex
ternal interrupt */
00287     gpioinitstruct.Mode   = GPIO_MODE_IT_FAL
LING;
00288     HAL_GPIO_Init(BUTTON_PORT[Button], &gpio
initstruct);
00289
00290     /* Enable and set Button EXTI Interrupt
to the lowest priority */
```

```
00291     HAL_NVIC_SetPriority((IRQn_Type)(BUTTON_
IRQn[Button]), 0x0F, 0);
00292     HAL_NVIC_EnableIRQ((IRQn_Type)(BUTTON_IR
Qn[Button]));
00293   }
00294 }
00295
00296 /**
00297   * @brief  Push Button DeInit.
00298   * @param  Button: Button to be configured
00299   *   This parameter should be: BUTTON_USER

00300   * @note PB DeInit does not disable the GPI
O clock
00301   * @retval None
00302   */
00303 void BSP_PB_DeInit(Button_TypeDef Button)
00304 {
00305   GPIO_InitTypeDef gpio_init_structure;
00306
00307   gpio_init_structure.Pin = BUTTON_PIN[Butto
n];
00308   HAL_NVIC_DisableIRQ((IRQn_Type)(BUTTON_IRQn
[Button]));
00309   HAL_GPIO_DeInit(BUTTON_PORT[Button], gpio_
init_structure.Pin);
00310 }
00311
00312 /**
00313   * @brief  Returns the selected Button stat
e.
00314   * @param  Button: Specifies the Button to
be checked.
00315   *   This parameter should be: BUTTON_USER
00316   * @retval Button state.
00317   */
00318 uint32_t BSP_PB_GetState(Button_TypeDef Butt
```

```
on)
00319 {
00320   return HAL_GPIO_ReadPin(BUTTON_PORT[Button
], BUTTON_PIN[Button]);
00321 }
00322
00323
00324 #ifdef HAL_SPI_MODULE_ENABLED
00325 /*****************************************
**********************************
00326                                     BUS OPERATIONS
00327 *****************************************
**********************************/
00328 /**
00329   * @brief  Initialize SPI MSP.
00330   * @retval None
00331   */
00332 static void SPIx_MspInit(void)
00333 {
00334   GPIO_InitTypeDef  gpioinitstruct = {0};
00335
00336   /*** Configure the GPIOs ***/
00337   /* Enable GPIO clock */
00338   NUCLEO_SPIx_SCK_GPIO_CLK_ENABLE();
00339   NUCLEO_SPIx_MISO_MOSI_GPIO_CLK_ENABLE();
00340
00341   /* Configure SPI SCK */
00342   gpioinitstruct.Pin = NUCLEO_SPIx_SCK_PIN;
00343   gpioinitstruct.Mode = GPIO_MODE_AF_PP;
00344   gpioinitstruct.Pull  = GPIO_PULLUP;
00345   gpioinitstruct.Speed = GPIO_SPEED_FREQ_VER
Y_HIGH;
00346   gpioinitstruct.Alternate = NUCLEO_SPIx_SCK
_AF;
00347   HAL_GPIO_Init(NUCLEO_SPIx_SCK_GPIO_PORT, &
gpioinitstruct);
00348
```

```
00349   /* Configure SPI MISO and MOSI */
00350   gpioinitstruct.Pin = NUCLEO_SPIx_MOSI_PIN;
00351   gpioinitstruct.Alternate = NUCLEO_SPIx_MIS
O_MOSI_AF;
00352   gpioinitstruct.Pull  = GPIO_PULLDOWN;
00353   HAL_GPIO_Init(NUCLEO_SPIx_MISO_MOSI_GPIO_P
ORT, &gpioinitstruct);
00354
00355   gpioinitstruct.Pin = NUCLEO_SPIx_MISO_PIN;
00356   HAL_GPIO_Init(NUCLEO_SPIx_MISO_MOSI_GPIO_P
ORT, &gpioinitstruct);
00357
00358   /*** Configure the SPI peripheral ***/
00359   /* Enable SPI clock */
00360   NUCLEO_SPIx_CLK_ENABLE();
00361 }
00362
00363 /**
00364   * @brief  Initialize SPI HAL.
00365   * @retval None
00366   */
00367 static void SPIx_Init(void)
00368 {
00369   if(HAL_SPI_GetState(&hnucleo_Spi) == HAL_S
PI_STATE_RESET)
00370   {
00371     /* SPI Config */
00372     hnucleo_Spi.Instance = NUCLEO_SPIx;
00373       /* SPI baudrate is set to 8 MHz maximu
m (PCLK2/SPI_BaudRatePrescaler = 32/4 = 8 MHz)
00374        to verify these constraints:
00375           - ST7735 LCD SPI interface max bau
drate is 15MHz for write and 6.66MHz for read
00376             Since the provided driver doesn'
t use read capability from LCD, only constraint
00377             on write baudrate is considered.
00378           - SD card SPI interface max baudra
```

```
te is 25MHz for write/read
00379            - PCLK2 max frequency is 32 MHz
00380         */
00381     hnucleo_Spi.Init.BaudRatePrescaler = SPI
_BAUDRATEPRESCALER_4;
00382     hnucleo_Spi.Init.Direction = SPI_DIRECTI
ON_2LINES;
00383     hnucleo_Spi.Init.CLKPhase = SPI_PHASE_1E
DGE;
00384     hnucleo_Spi.Init.CLKPolarity = SPI_POLAR
ITY_LOW;
00385     hnucleo_Spi.Init.CRCCalculation = SPI_CR
CCALCULATION_DISABLE;
00386     hnucleo_Spi.Init.CRCPolynomial = 7;
00387     hnucleo_Spi.Init.DataSize = SPI_DATASIZE
_8BIT;
00388     hnucleo_Spi.Init.FirstBit = SPI_FIRSTBIT
_MSB;
00389     hnucleo_Spi.Init.NSS = SPI_NSS_SOFT;
00390     hnucleo_Spi.Init.TIMode = SPI_TIMODE_DIS
ABLE;
00391     hnucleo_Spi.Init.Mode = SPI_MODE_MASTER;
00392
00393     SPIx_MspInit();
00394     HAL_SPI_Init(&hnucleo_Spi);
00395   }
00396 }
00397
00398 /**
00399   * @brief  SPI Write a byte to device
00400   * @param  DataIn: value to be written
00401   * @param  DataOut: read value
00402   * @param  DataLength: number of bytes to w
rite
00403   * @retval None
00404   */
00405 static void SPIx_WriteReadData(const uint8_t
```

```
 *DataIn, uint8_t *DataOut, uint16_t DataLength)
00406 {
00407   HAL_StatusTypeDef status = HAL_OK;
00408
00409   status = HAL_SPI_TransmitReceive(&hnucleo_
Spi, (uint8_t*) DataIn, DataOut, DataLength, SpixT
imeout);
00410
00411   /* Check the communication status */
00412   if(status != HAL_OK)
00413   {
00414     /* Execute user timeout callback */
00415     SPIx_Error();
00416   }
00417 }
00418
00419 /**
00420   * @brief  SPI Write an amount of data to d
evice
00421   * @param  DataIn: value to be written
00422   * @param  DataLength: number of bytes to w
rite
00423   * @retval None
00424   */
00425 static void SPIx_WriteData(uint8_t *DataIn,
uint16_t DataLength)
00426 {
00427   HAL_StatusTypeDef status = HAL_OK;
00428
00429   status = HAL_SPI_Transmit(&hnucleo_Spi, Da
taIn, DataLength, SpixTimeout);
00430
00431   /* Check the communication status */
00432   if(status != HAL_OK)
00433   {
00434     /* Execute user timeout callback */
00435     SPIx_Error();
```

```
00436   }
00437 }
00438
00439 /**
00440   * @brief  SPI Write a byte to device
00441   * @param  Value: value to be written
00442   * @retval None
00443   */
00444 static void SPIx_Write(uint8_t Value)
00445 {
00446   HAL_StatusTypeDef status = HAL_OK;
00447   uint8_t data;
00448
00449   status = HAL_SPI_TransmitReceive(&hnucleo_
Spi, (uint8_t*) &Value, &data, 1, SpixTimeout);
00450
00451   /* Check the communication status */
00452   if(status != HAL_OK)
00453   {
00454     /* Execute user timeout callback */
00455     SPIx_Error();
00456   }
00457 }
00458
00459 /**
00460   * @brief  SPI error treatment function
00461   * @retval None
00462   */
00463 static void SPIx_Error (void)
00464 {
00465   /* De-initialize the SPI communication BUS
 */
00466   HAL_SPI_DeInit(&hnucleo_Spi);
00467
00468   /* Re-Initiaize the SPI communication BUS
*/
00469   SPIx_Init();
```

```
00470 }
00471
00472 /*****************************************
*********************************
00473                                       LINK OPERATIONS
00474 *****************************************
*********************************/
00475
00476 /********************************* LINK SD *
*********************************/
00477 /**
00478   * @brief  Initialize the SD Card and put i
t into StandBy State (Ready for
00479   *         data transfer).
00480   * @retval None
00481   */
00482 void SD_IO_Init(void)
00483 {
00484   GPIO_InitTypeDef  gpioinitstruct = {0};
00485   uint8_t counter = 0;
00486
00487   /* SD_CS_GPIO Periph clock enable */
00488   SD_CS_GPIO_CLK_ENABLE();
00489
00490   /* Configure SD_CS_PIN pin: SD Card CS pin
 */
00491   gpioinitstruct.Pin = SD_CS_PIN;
00492   gpioinitstruct.Mode = GPIO_MODE_OUTPUT_PP;
00493   gpioinitstruct.Pull = GPIO_PULLUP;
00494   gpioinitstruct.Speed  = GPIO_SPEED_FREQ_HI
GH;
00495   HAL_GPIO_Init(SD_CS_GPIO_PORT, &gpioinitst
ruct);
00496
00497   /* Configure LCD_CS_PIN pin: LCD Card CS p
in */
00498   gpioinitstruct.Pin = LCD_CS_PIN;
```

```
00499   gpioinitstruct.Mode = GPIO_MODE_OUTPUT_PP;
00500   gpioinitstruct.Pull = GPIO_NOPULL;
00501   gpioinitstruct.Speed = GPIO_SPEED_FREQ_HIG
H;
00502   HAL_GPIO_Init(SD_CS_GPIO_PORT, &gpioinitst
ruct);
00503   LCD_CS_HIGH();
00504   /*------------Put SD in SPI mode----------
----*/
00505   /* SD SPI Config */
00506   SPIx_Init();
00507
00508   /* SD chip select high */
00509   SD_CS_HIGH();
00510
00511   /* Send dummy byte 0xFF, 10 times with CS
high */
00512   /* Rise CS and MOSI for 80 clocks cycles */

00513   for (counter = 0; counter <= 9; counter++)
00514   {
00515     /* Send dummy byte 0xFF */
00516     SD_IO_WriteByte(SD_DUMMY_BYTE);
00517   }
00518 }
00519
00520 /**
00521   * @brief  Set the SD_CS pin.
00522   * @param  val: pin value.
00523   * @retval None
00524   */
00525 void SD_IO_CSState(uint8_t val)
00526 {
00527   if(val == 1)
00528   {
00529     SD_CS_HIGH();
00530   }
```

```
00531   else
00532   {
00533     SD_CS_LOW();
00534   }
00535 }
00536
00537 /**
00538   * @brief  Write byte(s) on the SD
00539   * @param  DataIn: Pointer to data buffer t
o write
00540   * @param  DataOut: Pointer to data buffer
for read data
00541   * @param  DataLength: number of bytes to w
rite
00542   * @retval None
00543   */
00544 void SD_IO_WriteReadData(const uint8_t *Data
In, uint8_t *DataOut, uint16_t DataLength)
00545 {
00546   /* Send the byte */
00547   SPIx_WriteReadData(DataIn, DataOut, DataLe
ngth);
00548 }
00549
00550 /**
00551   * @brief  Write a byte on the SD.
00552   * @param  Data: byte to send.
00553   * @retval Data written
00554   */
00555 uint8_t SD_IO_WriteByte(uint8_t Data)
00556 {
00557   uint8_t tmp;
00558
00559   /* Send the byte */
00560   SPIx_WriteReadData(&Data,&tmp,1);
00561   return tmp;
00562 }
```

```
00563
00564 /**
00565   * @brief  Write an amount of data on the S
D.
00566   * @param  DataOut: byte to send.
00567   * @param  DataLength: number of bytes to w
rite
00568   * @retval none
00569   */
00570 void SD_IO_ReadData(uint8_t *DataOut, uint16
_t DataLength)
00571 {
00572   /* Send the byte */
00573   SD_IO_WriteReadData(DataOut, DataOut, Data
Length);
00574   }
00575
00576 /**
00577   * @brief  Write an amount of data on the S
D.
00578   * @param  Data: byte to send.
00579   * @param  DataLength: number of bytes to w
rite
00580   * @retval none
00581   */
00582 void SD_IO_WriteData(const uint8_t *Data, ui
nt16_t DataLength)
00583 {
00584   /* Send the byte */
00585   SPIx_WriteData((uint8_t *)Data, DataLength
);
00586 }
00587
00588 /******************************** LINK LCD
********************************/
00589 /**
00590   * @brief  Initialize the LCD
```

```
00591   * @retval None
00592   */
00593 void LCD_IO_Init(void)
00594 {
00595   GPIO_InitTypeDef  gpioinitstruct = {0};
00596
00597   /* LCD_CS_GPIO and LCD_DC_GPIO Periph cloc
k enable */
00598   LCD_CS_GPIO_CLK_ENABLE();
00599   LCD_DC_GPIO_CLK_ENABLE();
00600
00601   /* Configure LCD_CS_PIN pin: LCD Card CS p
in */
00602   gpioinitstruct.Pin = LCD_CS_PIN;
00603   gpioinitstruct.Mode = GPIO_MODE_OUTPUT_PP;
00604   gpioinitstruct.Pull = GPIO_NOPULL;
00605   gpioinitstruct.Speed = GPIO_SPEED_FREQ_VER
Y_HIGH;
00606   HAL_GPIO_Init(SD_CS_GPIO_PORT, &gpioinitst
ruct);
00607
00608   /* Configure LCD_DC_PIN pin: LCD Card DC p
in */
00609   gpioinitstruct.Pin = LCD_DC_PIN;
00610   HAL_GPIO_Init(LCD_DC_GPIO_PORT, &gpioinits
truct);
00611
00612   /* LCD chip select high */
00613   LCD_CS_HIGH();
00614
00615   /* LCD SPI Config */
00616   SPIx_Init();
00617 }
00618
00619 /**
00620   * @brief  Write command to select the LCD
register.
```

```
00621   * @param  LCDReg: Address of the selected
register.
00622   * @retval None
00623   */
00624 void LCD_IO_WriteReg(uint8_t LCDReg)
00625 {
00626   /* Reset LCD control line CS */
00627   LCD_CS_LOW();
00628
00629   /* Set LCD data/command line DC to Low */
00630   LCD_DC_LOW();
00631
00632   /* Send Command */
00633   SPIx_Write(LCDReg);
00634
00635   /* Deselect : Chip Select high */
00636   LCD_CS_HIGH();
00637 }
00638
00639 /**
00640   * @brief  Writes data to select the LCD re
gister.
00641   *         This function must be used after
 st7735_WriteReg() function
00642   * @param  Data: data to write to the selec
ted register.
00643   * @retval None
00644   */
00645 void LCD_IO_WriteData(uint8_t Data)
00646 {
00647   /* Reset LCD control line CS */
00648   LCD_CS_LOW();
00649
00650   /* Set LCD data/command line DC to High */
00651   LCD_DC_HIGH();
00652
00653   /* Send Data */
```

```
00654   SPIx_Write(Data);
00655
00656   /* Deselect : Chip Select high */
00657   LCD_CS_HIGH();
00658 }
00659
00660 /**
00661 * @brief  Write register value.
00662 * @param  pData Pointer on the register value

00663 * @param  Size Size of byte to transmit to t
he register
00664 * @retval None
00665 */
00666 void LCD_IO_WriteMultipleData(uint8_t *pData
, uint32_t Size)
00667 {
00668   uint32_t counter = 0;
00669   __IO uint32_t data = 0;
00670
00671   /* Reset LCD control line CS */
00672   LCD_CS_LOW();
00673
00674   /* Set LCD data/command line DC to High */
00675   LCD_DC_HIGH();
00676
00677   if (Size == 1)
00678   {
00679     /* Only 1 byte to be sent to LCD - gener
al interface can be used */
00680     /* Send Data */
00681     SPIx_Write(*pData);
00682   }
00683   else
00684   {
00685     /* Several data should be sent in a raw
*/
```

```
00686     /* Direct SPI accesses for optimization
*/
00687     for (counter = Size; counter != 0; count
er--)
00688     {
00689       while(((hnucleo_Spi.Instance->SR) & SP
I_FLAG_TXE) != SPI_FLAG_TXE)
00690       {
00691       }
00692       /* Need to invert bytes for LCD*/
00693       *((__IO uint8_t*)&hnucleo_Spi.Instance
->DR) = *(pData+1);
00694
00695       while(((hnucleo_Spi.Instance->SR) & SP
I_FLAG_TXE) != SPI_FLAG_TXE)
00696       {
00697       }
00698       *((__IO uint8_t*)&hnucleo_Spi.Instance
->DR) = *pData;
00699       counter--;
00700       pData += 2;
00701     }
00702
00703     /* Wait until the bus is ready before re
leasing Chip select */
00704     while(((hnucleo_Spi.Instance->SR) & SPI_
FLAG_BSY) != RESET)
00705     {
00706     }
00707   }
00708
00709   /* Empty the Rx fifo */
00710   data = *(&hnucleo_Spi.Instance->DR);
00711   UNUSED(data);  /* Remove GNU warning */
00712
00713   /* Deselect : Chip Select high */
00714   LCD_CS_HIGH();
```

```
00715 }
00716
00717 /**
00718   * @brief  Wait for loop in ms.
00719   * @param  Delay in ms.
00720   * @retval None
00721   */
00722 void LCD_Delay(uint32_t Delay)
00723 {
00724   HAL_Delay(Delay);
00725 }
00726 #endif /* HAL_SPI_MODULE_ENABLED */
00727
00728 /****************************** LINK JOYSTI
CK *****************************/
00729 #ifdef HAL_ADC_MODULE_ENABLED
00730 /**
00731   * @brief  Initialize ADC MSP.
00732   * @retval None
00733   */
00734 static void ADCx_MspInit(ADC_HandleTypeDef *
hadc)
00735 {
00736   GPIO_InitTypeDef  gpioinitstruct = {0};
00737
00738   /*** Configure the GPIOs ***/
00739   /* Enable GPIO clock */
00740   NUCLEO_ADCx_GPIO_CLK_ENABLE();
00741
00742   /* Configure ADC1 Channel8 as analog input
 */
00743   gpioinitstruct.Pin = NUCLEO_ADCx_GPIO_PIN
;
00744   gpioinitstruct.Mode = GPIO_MODE_ANALOG;
00745   gpioinitstruct.Pull   = GPIO_NOPULL;
00746   HAL_GPIO_Init(NUCLEO_ADCx_GPIO_PORT, &gpio
initstruct);
```

```
00747
00748   /*** Configure the ADC peripheral ***/
00749   /* Enable ADC clock */
00750   NUCLEO_ADCx_CLK_ENABLE();
00751 }
00752
00753 /**
00754   * @brief  DeInitializes ADC MSP.
00755   * @param  hadc: ADC peripheral
00756   * @note ADC DeInit does not disable the GP
IO clock
00757   * @retval None
00758   */
00759 static void ADCx_MspDeInit(ADC_HandleTypeDef
 *hadc)
00760 {
00761   GPIO_InitTypeDef  gpioinitstruct;
00762
00763   /*** DeInit the ADC peripheral ***/
00764   /* Disable ADC clock */
00765   NUCLEO_ADCx_CLK_DISABLE();
00766
00767   /* Configure the selected ADC Channel as a
nalog input */
00768   gpioinitstruct.Pin = NUCLEO_ADCx_GPIO_PIN
;
00769   HAL_GPIO_DeInit(NUCLEO_ADCx_GPIO_PORT, gpi
oinitstruct.Pin);
00770
00771   /* Disable GPIO clock has to be done by th
e application*/
00772   /* NUCLEO_ADCx_GPIO_CLK_DISABLE(); */
00773 }
00774
00775 /**
00776   * @brief  Initializes ADC HAL.
00777   * @retval None
```

```
00778   */
00779 static HAL_StatusTypeDef ADCx_Init(void)
00780 {
00781   /* Set ADC instance */
00782   hnucleo_Adc.Instance = NUCLEO_ADCx;
00783
00784   if(HAL_ADC_GetState(&hnucleo_Adc) == HAL_A
DC_STATE_RESET)
00785   {
00786     /* ADC Config */
00787     hnucleo_Adc.Instance = NUCLEO_ADCx;
00788     hnucleo_Adc.Init.OversamplingMode      =
 DISABLE;
00789     hnucleo_Adc.Init.ClockPrescaler        =
 ADC_CLOCK_SYNC_PCLK_DIV2; /* (must not exceed 16M
Hz) */
00790     hnucleo_Adc.Init.LowPowerAutoPowerOff  =
 DISABLE;
00791     hnucleo_Adc.Init.LowPowerFrequencyMode =
 ENABLE;
00792     hnucleo_Adc.Init.LowPowerAutoWait      =
 ENABLE;
00793     hnucleo_Adc.Init.Resolution            =
 ADC_RESOLUTION_12B;
00794     hnucleo_Adc.Init.SamplingTime          =
 ADC_SAMPLETIME_1CYCLE_5;
00795     hnucleo_Adc.Init.ScanConvMode          =
 ADC_SCAN_DIRECTION_FORWARD;
00796     hnucleo_Adc.Init.DataAlign             =
 ADC_DATAALIGN_RIGHT;
00797     hnucleo_Adc.Init.ContinuousConvMode    =
 DISABLE;
00798     hnucleo_Adc.Init.DiscontinuousConvMode =
 DISABLE;
00799     hnucleo_Adc.Init.ExternalTrigConv
= ADC_SOFTWARE_START;           /* Trig of conver
sion start done manually by software, without exte
```

```
rnal event */
00800     hnucleo_Adc.Init.ExternalTrigConvEdge
= ADC_EXTERNALTRIGCONVEDGE_NONE; /* Parameter disc
arded because trig by software start */
00801     hnucleo_Adc.Init.EOCSelection
= ADC_EOC_SEQ_CONV;
00802     hnucleo_Adc.Init.DMAContinuousRequests
= DISABLE;
00803
00804     /* Initialize MSP related to ADC */
00805     ADCx_MspInit(&hnucleo_Adc);
00806
00807     /* Initialize ADC */
00808     if (HAL_ADC_Init(&hnucleo_Adc) != HAL_OK
)
00809     {
00810       return HAL_ERROR;
00811     }
00812
00813     if (HAL_ADCEx_Calibration_Start(&hnucleo
_Adc,ADC_SINGLE_ENDED) != HAL_OK)
00814     {
00815       return HAL_ERROR;
00816     }
00817   }
00818
00819   return HAL_OK;
00820 }
00821
00822 /**
00823   * @brief  Initializes ADC HAL.
00824   * @retval None
00825   */
00826 static void ADCx_DeInit(void)
00827 {
00828     hnucleo_Adc.Instance   = NUCLEO_ADCx;
00829
```

```
00830     HAL_ADC_DeInit(&hnucleo_Adc);
00831     ADCx_MspDeInit(&hnucleo_Adc);
00832 }
00833
00834 /****************************** LINK JOYSTI
CK *****************************/
00835
00836 /**
00837   * @brief  Configures joystick available on
 adafruit 1.8" TFT shield
00838   *         managed through ADC to detect mo
tion.
00839   * @retval Joystickstatus (0=> success, 1=>
 fail)
00840   */
00841 uint8_t BSP_JOY_Init(void)
00842 {
00843   if (ADCx_Init() != HAL_OK)
00844   {
00845     return (uint8_t) HAL_ERROR;
00846   }
00847
00848   /* Select Channel 8 to be converted */
00849   sConfig.Channel = ADC_CHANNEL_8;
00850   sConfig.Rank = ADC_RANK_CHANNEL_NUMBER;
00851
00852   /* Return Joystick initialization status */

00853   return (uint8_t) HAL_ADC_ConfigChannel(&hn
ucleo_Adc, &sConfig);
00854 }
00855
00856 /**
00857   * @brief  DeInit joystick GPIOs.
00858   * @note   JOY DeInit does not disable the
Mfx, just set the Mfx pins in Off mode
00859   * @retval None.
```

```
00860   */
00861 void BSP_JOY_DeInit(void)
00862 {
00863     ADCx_DeInit();
00864 }
00865
00866 /**
00867   * @brief  Returns the Joystick key pressed.

00868   * @note   To know which Joystick key is pr
essed we need to detect the voltage
00869   *         level on each key output
00870   *           - None  : 3.3 V / 4095
00871   *           - SEL   : 1.055 V / 1308
00872   *           - DOWN  : 0.71 V / 88
00873   *           - LEFT  : 3.0 V / 3720
00874   *           - RIGHT : 0.595 V / 737
00875   *           - UP    : 1.65 V / 2046
00876   * @retval JOYState_TypeDef: Code of the Jo
ystick key pressed.
00877   */
00878 JOYState_TypeDef BSP_JOY_GetState(void)
00879 {
00880   JOYState_TypeDef state = JOY_NONE;
00881   uint16_t  keyconvertedvalue = 0;
00882
00883  /* Start the conversion process */
00884   HAL_ADC_Start(&hnucleo_Adc);
00885
00886   /* Wait for the end of conversion */
00887   if (HAL_ADC_PollForConversion(&hnucleo_Adc
, 10) != HAL_TIMEOUT)
00888   {
00889     /* Get the converted value of regular ch
annel */
00890     keyconvertedvalue = HAL_ADC_GetValue(&hn
ucleo_Adc);
```

```
00891   }
00892
00893   if((keyconvertedvalue > 2010) && (keyconve
rtedvalue < 2090))
00894   {
00895     state = JOY_UP;
00896   }
00897   else if((keyconvertedvalue > 680) && (keyc
onvertedvalue < 780))
00898   {
00899     state = JOY_RIGHT;
00900   }
00901   else if((keyconvertedvalue > 1270) && (key
convertedvalue < 1350))
00902   {
00903     state = JOY_SEL;
00904   }
00905   else if((keyconvertedvalue > 50) && (keyco
nvertedvalue < 130))
00906   {
00907     state = JOY_DOWN;
00908   }
00909   else if((keyconvertedvalue > 3570) && (key
convertedvalue < 3800))
00910   {
00911     state = JOY_LEFT;
00912   }
00913   else
00914   {
00915     state = JOY_NONE;
00916   }
00917
00918   /* Return the code of the Joystick key pre
ssed */
00919   return state;
00920 }
00921 #endif /* HAL_ADC_MODULE_ENABLED */
```

```
00922
00923 /**
00924   * @}
00925   */
00926
00927 /**
00928   * @}
00929   */
00930
00931 /**
00932   * @}
00933   */
00934
00935 /**
00936   * @}
00937   */
00938
00939 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32L0xx_Nucleo BSP User Manual

BSP

Modules

- **STM32L0XX_NUCLEO**

STM32L0xx_Nucleo BSP User Manual

STM32L0XX_NUCLEO

BSP

Modules

STM32L0XX_NUCLEO_LOW_LEVEL

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