STM32F407VG Device Drivers

Generated by Doxygen 1.9.1

Author: Mr. Mohamed Ali Haoufa

Documentation version: 1.0

Date: October 7th, 2023

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

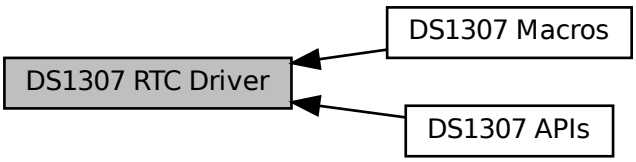
Chapter 4

Module Documentation

4.1 DS1307 RTC Driver

DS1307 Real-Time Clock (RTC) driver for STM32F4xx MCUs.

Collaboration diagram for DS1307 RTC Driver:



Modules

- DS1307 Macros

  Macros for DS1307 configuration and settings.
- DS1307 APIs

  APIs supported by the DS1307 driver.

Classes

- struct RTC_date_t

  Data structure for holding date information.
- struct RTC_time_t

  Data structure for holding time information.

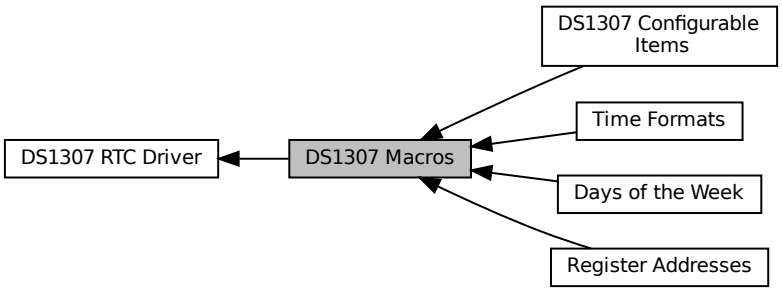
4.1.1 Detailed Description

DS1307 Real-Time Clock (RTC) driver for STM32F4xx MCUs.

4.2 DS1307 Macros

Macros for DS1307 configuration and settings.

Collaboration diagram for DS1307 Macros:



Modules

- DS1307 Configurable Items

  Macros for DS1307 configuration settings.
- Register Addresses

  Macros for DS1307 register addresses as per DS1307's Data-sheet.
- Time Formats

  Macros for DS1307 time format settings.
- Days of the Week

  Macros for DS1307 day of the week settings.

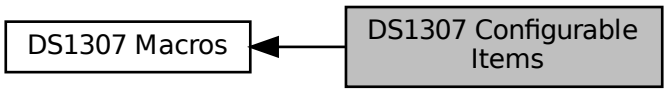
4.2.1 Detailed Description

Macros for DS1307 configuration and settings.

4.3 DS1307 Configurable Items

Macros for DS1307 configuration settings.

Collaboration diagram for DS1307 Configurable Items:



Macros

- #define **DS1307_I2C** I2C1
- #define **DS1307_I2C_GPIO_PORT** GPIOB
- #define **DS1307_I2C_SDA_PIN** GPIO_PIN_NO_7
- #define **DS1307_I2C_SCL_PIN** GPIO_PIN_NO_6
- #define **DS1307_I2C_SPEED** I2C_SC_SPEED_SM
- #define **DS1307_I2C_PUPD** GPIO_PIN_PU
- #define **DS1307_I2C_ADDRESS** 0x68

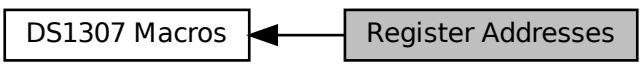
4.3.1 Detailed Description

Macros for DS1307 configuration settings.

4.4 Register Addresses

Macros for DS1307 register addresses as per DS1307's Data-sheet.

Collaboration diagram for Register Addresses:



Macros

- #define **DS1307_ADDR_SEC** 0x00
- #define **DS1307_ADDR_MIN** 0x01
- #define **DS1307_ADDR_HRS** 0x02
- #define **DS1307_ADDR_DAY** 0x03
- #define **DS1307_ADDR_DATE** 0x04
- #define **DS1307_ADDR_MONTH** 0x05
- #define **DS1307_ADDR_YEAR** 0x06

4.4.1 Detailed Description

Macros for DS1307 register addresses as per DS1307's Data-sheet.

4.5 Time Formats

Macros for DS1307 time format settings.

Collaboration diagram for Time Formats:

Macros

- #define **TIME_FORMAT_12HRS_AM** 0
- #define **TIME_FORMAT_12HRS_PM** 1
- #define **TIME_FORMAT_24HRS** 2

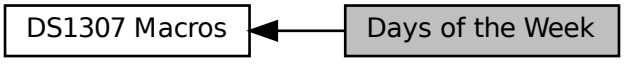
4.5.1 Detailed Description

Macros for DS1307 time format settings.

4.6 Days of the Week

Macros for DS1307 day of the week settings.

Collaboration diagram for Days of the Week:



Macros

- #define **SUNDAY** 1
- #define **MONDAY** 2
- #define **TUESDAY** 3
- #define **WEDNESDAY** 4
- #define **THURSDAY** 5
- #define **FRIDAY** 6
- #define **SATURDAY** 7

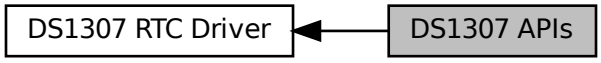
4.6.1 Detailed Description

Macros for DS1307 day of the week settings.

4.7 DS1307 APIs

APIs supported by the DS1307 driver.

Collaboration diagram for DS1307 APIs:



Functions

- uint8_t ds1307_init ()

  Initialize the DS1307 module.
- void ds1307_set_current_time (RTC_time_t *)

  Set the current time on the DS1307 RTC module.
- void ds1307_get_current_time (RTC_time_t *)

  Get the current time from the DS1307 RTC module.
- void ds1307_set_current_date (RTC_date_t *)

  Set the current date on the DS1307 RTC module.
- void ds1307_get_current_date (RTC_date_t *)

  Get the current date from the DS1307 RTC module.

4.7.1 Detailed Description

APIs supported by the DS1307 driver.

4.7.2 Function Documentation

4.7.2.1 ds1307_get_current_date()

```
void ds1307_get_current_date (
            RTC_date_t * rtc_date )
```

Get the current date from the DS1307 RTC module.

Parameters

out	*Pointer*	to a structure to store the retrieved date.

4.7.2.2 ds1307_get_current_time()

```
void ds1307_get_current_time (
            RTC_time_t * rtc_time )
```

Get the current time from the DS1307 RTC module.

Parameters

out	*Pointer*	to a structure to store the retrieved time.

4.7.2.3 ds1307_init()

```
uint8_t ds1307_init ( )
```

Initialize the DS1307 module.

Returns

Status:

- 0: Success
- 1: Error

4.7.2.4 ds1307_set_current_date()

```
void ds1307_set_current_date (
            RTC_date_t * rtc_date )
```

Set the current date on the DS1307 RTC module.

Parameters

in	*Pointer*	to a structure containing the date to set.

4.7.2.5 ds1307_set_current_time()

```
void ds1307_set_current_time (
            RTC_time_t * rtc_time )
```

Set the current time on the DS1307 RTC module.

Parameters

in	*Pointer*	to a structure containing the time to set.

4.8 KEYPAD Driver

Driver for interfacing with a keypad.

Functions

- void Keypad_ScanAndPrint (uint8_t row_number, uint8_t row_pin, uint8_t column_pin, GPIO_RegDef_t *p↩ GPIOx)

  Scan and print the pressed key on the keypad.

4.8.1 Detailed Description

Driver for interfacing with a keypad.

4.8.2 Function Documentation

4.8.2.1 Keypad_ScanAndPrint()

```
void Keypad_ScanAndPrint (
            uint8_t row_number,
            uint8_t row_pin,
            uint8_t column_pin,
            GPIO_RegDef_t * InputOutput_port )
```

Scan and print the pressed key on the keypad.

This function scans a specific row of the keypad for keypress and prints the pressed key on an LCD.

Parameters

in	*row_number*	The row number to scan (0 to 3 for a 4x4 keypad).
in	*row_pin*	The GPIO pin number connected to the row.
in	*column_pin*	The GPIO port connected to the column.
in	*pGPIOx*	Pointer to the GPIO peripheral associated with the keypad.

This function scans a specific row of the keypad for keypress and prints the pressed key on an LCD.

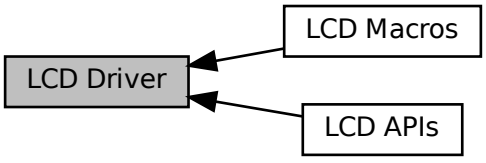
Parameters

in	*row_number*	The row number to scan (0 to 3 for a 4x4 keypad).
in	*row_pin*	The GPIO pin number connected to the row.
in	*column_pin*	The GPIO port connected to the column.
in	*InputOutput_port*	Pointer to the GPIO peripheral associated with the keypad.

4.9 LCD Driver

LCD (Liquid Crystal Display) driver for STM32F4xx MCUs.

Collaboration diagram for LCD Driver:



Modules

- LCD Macros

  Macros for LCD configuration and settings.
- LCD APIs

  APIs for interfacing with an LCD (Liquid Crystal Display).

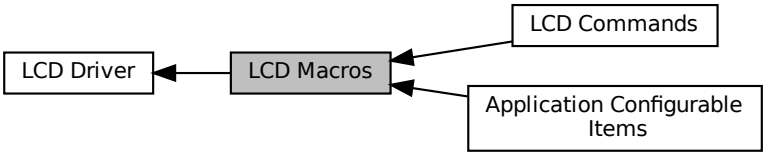
4.9.1 Detailed Description

LCD (Liquid Crystal Display) driver for STM32F4xx MCUs.

4.10 LCD Macros

Macros for LCD configuration and settings.

Collaboration diagram for LCD Macros:



Modules

- Application Configurable Items

  Macros for LCD configuration settings.
- LCD Commands

  Macros for LCD command codes.

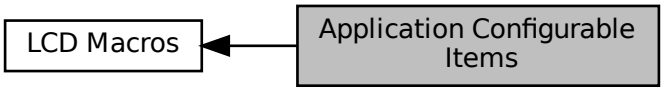
4.10.1 Detailed Description

Macros for LCD configuration and settings.

4.11 Application Configurable Items

Macros for LCD configuration settings.

Collaboration diagram for Application Configurable Items:



Macros

- #define **LCD_GPIO_PORT** GPIOD
- #define **LCD_GPIO_RS** GPIO_PIN_NO_0
- #define **LCD_GPIO_RW** GPIO_PIN_NO_1
- #define **LCD_GPIO_EN** GPIO_PIN_NO_2
- #define **LCD_GPIO_D4** GPIO_PIN_NO_3
- #define **LCD_GPIO_D5** GPIO_PIN_NO_4
- #define **LCD_GPIO_D6** GPIO_PIN_NO_5
- #define **LCD_GPIO_D7** GPIO_PIN_NO_6

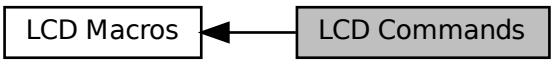
4.11.1 Detailed Description

Macros for LCD configuration settings.

4.12 LCD Commands

Macros for LCD command codes.

Collaboration diagram for LCD Commands:



Macros

- #define **LCD_CMD_4DL_2N_5X8F** 0x28
- #define **LCD_CMD_DON_CON** 0x0E
- #define **LCD_CMD_DIS_CLEAR** 0x01
- #define **LCD_CMD_INCADD** 0x06
- #define **LCD_CMD_DIS_RETURN_HOME** 0x02

4.12.1 Detailed Description

Macros for LCD command codes.

4.13 LCD APIs

APIs for interfacing with an LCD (Liquid Crystal Display).

Collaboration diagram for LCD APIs:

Functions

- void lcd_init (void)
  Initialize the LCD module.
- void lcd_send_command (uint8_t cmd)
  Send a command to the LCD.
- void lcd_print_char (uint8_t data)
  Print a character on the LCD.
- void lcd_print_string (char ∗message)
  Print a string on the LCD.
- void lcd_set_cursor (uint8_t row, uint8_t column)
  Set the cursor position on the LCD.
- void lcd_display_clear (void)
  Clear the LCD display.
- void lcd_display_return_home (void)
  Return the cursor to the home position on the LCD.

4.13.1 Detailed Description

APIs for interfacing with an LCD (Liquid Crystal Display).

4.13.2 Function Documentation

4.13.2.1 lcd_display_clear()

```
void lcd_display_clear (
            void  )
```

Clear the LCD display.

This function clears the entire content displayed on the LCD.

4.13.2.2 lcd_display_return_home()

```
void lcd_display_return_home (
            void  )
```

Return the cursor to the home position on the LCD.

This function returns the cursor to the top-left (home) position on the LCD screen.

4.13.2.3 lcd_init()

```
void lcd_init (
            void  )
```

Initialize the LCD module.

This function initializes the LCD module and prepares it for use.

4.13.2.4 lcd_print_char()

```
void lcd_print_char (
            uint8_t data )
```

Print a character on the LCD.

This function displays a single character on the LCD.

Parameters

in	*data*	The character to be displayed.

This function displays a single character on the LCD. Here we used 4 bit parallel data transmission. First higher nibble of the data will be sent on to the data lines D4, D5, D6, D7 Then lower nibble of the data will be set on to the data lines D4, D5, D6, D7

Parameters

in	*data*	The character to be displayed.

4.13.2.5 lcd_print_string()

```
void lcd_print_string (
            char * message )
```

Print a string on the LCD.

This function displays a null-terminated string on the LCD.

Parameters

in	*message*	Pointer to the null-terminated string to be displayed.

4.13.2.6 lcd_send_command()

```
void lcd_send_command (
            uint8_t cmd )
```

Send a command to the LCD.

This function sends a command to the LCD module for various control operations.

Parameters

in	*cmd*	The command to be sent.

4.13.2.7 lcd_set_cursor()

```
void lcd_set_cursor (
            uint8_t row,
            uint8_t column )
```

Set the cursor position on the LCD.

This function sets the cursor position on the LCD screen.

Parameters

in	*row*	The row (line) where the cursor should be placed (0 or 1).
in	*column*	The column where the cursor should be placed (0 to 15).

This function sets the cursor position on the LCD screen. Set Lcd to a specified location given by row and column information (page 11 in data-sheet) Row Number (1 to 2) Column Number (1 to 16) Assuming a 2 X 16 characters display

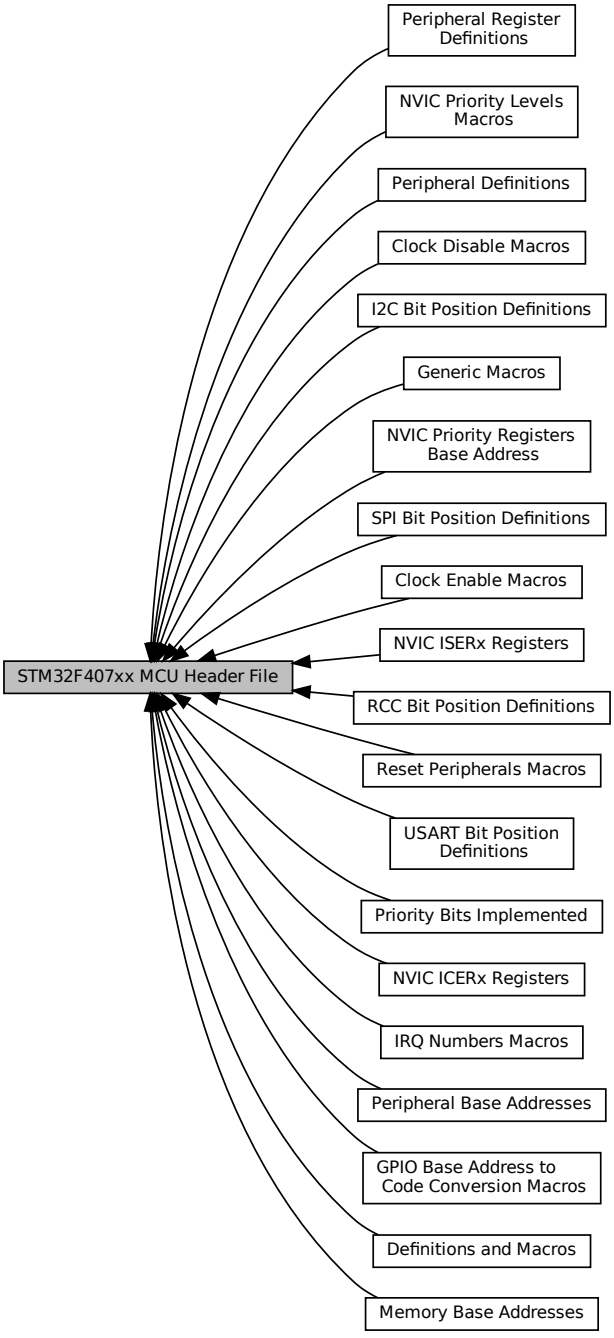
Parameters

in	*row*	The row (line) where the cursor should be placed (0 or 1).
in	*column*	The column where the cursor should be placed (0 to 15).

4.14 STM32F407xx MCU Header File

Header file containing all the necessary information about the STM32F407xx MCU.

Collaboration diagram for STM32F407xx MCU Header File:



Modules

- Definitions and Macros

Various definitions and macros for the STM32F407xx MCU.

- NVIC ISERx Registers

  Base addresses for NVIC ISERx registers.
- NVIC ICERx Registers

  Base addresses for NVIC ICERx registers.
- NVIC Priority Registers Base Address

  Base address for NVIC Priority registers calculation.
- Priority Bits Implemented

  Number of Priority bits implemented in the priority register.
- Memory Base Addresses

  Base addresses of memory regions.
- Peripheral Base Addresses

  Base addresses of various peripherals for the STM32F407xx MCU.
- Peripheral Register Definitions

  Structures defining the register layouts for various peripherals.
- Peripheral Definitions

  Definitions for various peripheral instances based on their base addresses.
- Clock Enable Macros
- Clock Disable Macros
- Reset Peripherals Macros

  Macros for resetting various peripherals.
- GPIO Base Address to Code Conversion Macros

  Macros for converting GPIO base addresses to corresponding port codes.
- IRQ Numbers Macros

  Macros for interrupt request numbers and priority levels.
- NVIC Priority Levels Macros

  Macros for all possible priority levels for NVIC.
- SPI Bit Position Definitions

  Bit position definitions for various registers in the SPI peripheral.
- I2C Bit Position Definitions

  Bit position definitions for various registers in the I2C peripheral.
- USART Bit Position Definitions

  Bit position definitions for various registers in the USART peripheral.
- RCC Bit Position Definitions

  Bit position definitions for various registers in the RCC peripheral.
- Generic Macros

  Generic macros for enabling/disabling, setting/resetting, and handling flags.

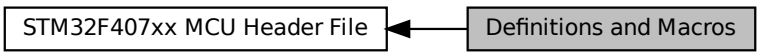
4.14.1 Detailed Description

Header file containing all the necessary information about the STM32F407xx MCU.

4.15 Definitions and Macros

Various definitions and macros for the STM32F407xx MCU.

Collaboration diagram for Definitions and Macros:



Macros

- #define __vo volatile

  volatile_32bit_register Define for accessing a volatile 32-bit register
- #define __weak __attribute__((weak))

  Define for the weak attribute.

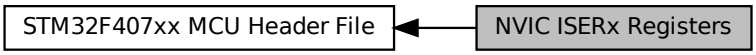
4.15.1 Detailed Description

Various definitions and macros for the STM32F407xx MCU.

4.16 NVIC ISERx Registers

Base addresses for NVIC ISERx registers.

Collaboration diagram for NVIC ISERx Registers:



Macros

- #define **NVIC_ISER0** ((__vo uint32_t*)0xE000E100)
- #define **NVIC_ISER1** ((__vo uint32_t*)0xE000E104)
- #define **NVIC_ISER2** ((__vo uint32_t*)0xE000E108)
- #define **NVIC_ISER3** ((__vo uint32_t*)0xE000E10C)
- #define **NVIC_ISER4** ((__vo uint32_t*)0xE000E110)
- #define **NVIC_ISER5** ((__vo uint32_t*)0xE000E114)
- #define **NVIC_ISER6** ((__vo uint32_t*)0xE000E118)
- #define **NVIC_ISER7** ((__vo uint32_t*)0xE000E11C)

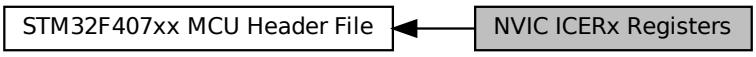
4.16.1 Detailed Description

Base addresses for NVIC ISERx registers.

4.17 NVIC ICERx Registers

Base addresses for NVIC ICERx registers.

Collaboration diagram for NVIC ICERx Registers:



Macros

- #define **NVIC_ICER0** ((__vo uint32_t*)0xE000E180)
- #define **NVIC_ICER1** ((__vo uint32_t*)0xE000E184)
- #define **NVIC_ICER2** ((__vo uint32_t*)0xE000E188)
- #define **NVIC_ICER3** ((__vo uint32_t*)0xE000E18C)
- #define **NVIC_ICER4** ((__vo uint32_t*)0xE000E190)
- #define **NVIC_ICER5** ((__vo uint32_t*)0xE000E194)
- #define **NVIC_ICER6** ((__vo uint32_t*)0xE000E198)
- #define **NVIC_ICER7** ((__vo uint32_t*)0xE000E19C)

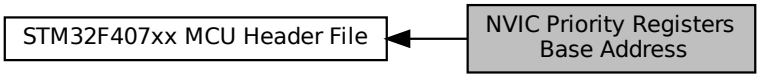
4.17.1 Detailed Description

Base addresses for NVIC ICERx registers.

4.18 NVIC Priority Registers Base Address

Base address for NVIC Priority registers calculation.

Collaboration diagram for NVIC Priority Registers Base Address:



Macros

- #define **NVIC_PR_BASE_ADDR** ((__vo uint32_t*)0xE000E400)

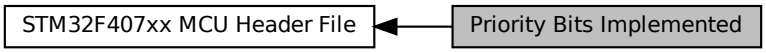
4.18.1 Detailed Description

Base address for NVIC Priority registers calculation.

4.19 Priority Bits Implemented

Number of Priority bits implemented in the priority register.

Collaboration diagram for Priority Bits Implemented:



Macros

- #define **NO_PR_BITS_IMPLEMENTED** 4

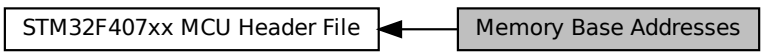
4.19.1 Detailed Description

Number of Priority bits implemented in the priority register.

4.20 Memory Base Addresses

Base addresses of memory regions.

Collaboration diagram for Memory Base Addresses:



Macros

- #define FLASH_BASEADDR 0x08000000U
- #define SRAM1_BASEADDR (uint32_t)0x20000000
- #define **SRAM** SRAM1_BASE_ADDR
- #define ROM_BASEADDR 0x1FFF0000U
- #define SRAM2_BASEADDR 0x2001C000U

4.20.1 Detailed Description

Base addresses of memory regions.

4.20.2 Macro Definition Documentation

4.20.2.1 FLASH_BASEADDR

```
#define FLASH_BASEADDR 0x08000000U
```

Base address of FLASH memory

4.20.2.2 ROM_BASEADDR

```
#define ROM_BASEADDR 0x1FFF0000U
```

Base address of ROM memory

4.20.2.3 SRAM1_BASEADDR

```
#define SRAM1_BASEADDR (uint32_t)0x20000000
```

Base address of SRAM1 memory

4.20.2.4 SRAM2_BASEADDR

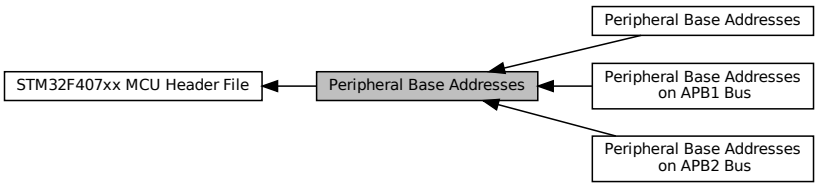
```
#define SRAM2_BASEADDR 0x2001C000U
```

Base address of SRAM2 memory

4.21 Peripheral Base Addresses

Base addresses of various peripherals for the STM32F407xx MCU.

Collaboration diagram for Peripheral Base Addresses:



Modules

- Peripheral Base Addresses

  Base addresses of AHBx and APBx bus peripherals.
- Peripheral Base Addresses on APB1 Bus

  Base addresses of peripherals connected to the APB1 bus.
- Peripheral Base Addresses on APB2 Bus

  Base addresses of peripherals connected to the APB2 bus.

4.21.1 Detailed Description

Base addresses of various peripherals for the STM32F407xx MCU.

4.22 Peripheral Base Addresses

Base addresses of AHBx and APBx bus peripherals.

Collaboration diagram for Peripheral Base Addresses:

Macros

- #define PERIPH_BASEADDR 0x40000000U
- #define APB1PERIPH_BASEADDR PERIPH_BASEADDR
- #define APB2PERIPH_BASEADDR 0x40010000U
- #define AHB1PERIPH_BASEADDR 0x40020000U
- #define AHB2PERIPH_BASEADDR 0x50000000U

4.22.1 Detailed Description

Base addresses of AHBx and APBx bus peripherals.

4.22.2 Macro Definition Documentation

4.22.2.1 AHB1PERIPH_BASEADDR

```
#define AHB1PERIPH_BASEADDR 0x40020000U
```

Base address of AHB1 peripheral memory

4.22.2.2 AHB2PERIPH_BASEADDR

```
#define AHB2PERIPH_BASEADDR 0x50000000U
```

Base address of AHB2 peripheral memory

4.22.2.3 APB1PERIPH_BASEADDR

```
#define APB1PERIPH_BASEADDR PERIPH_BASEADDR
```

Base address of APB1 peripheral memory

4.22.2.4 APB2PERIPH_BASEADDR

```
#define APB2PERIPH_BASEADDR 0x40010000U
```

Base address of APB2 peripheral memory

4.22.2.5 PERIPH_BASEADDR

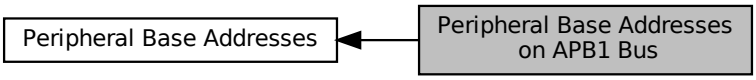
```
#define PERIPH_BASEADDR 0x40000000U
```

Base address of peripheral memory

4.23 Peripheral Base Addresses on APB1 Bus

Base addresses of peripherals connected to the APB1 bus.

Collaboration diagram for Peripheral Base Addresses on APB1 Bus:



Macros

- #define GPIOA_BASEADDR (AHB1PERIPH_BASEADDR + 0x0000)
- #define GPIOB_BASEADDR (AHB1PERIPH_BASEADDR + 0x0400)
- #define GPIOC_BASEADDR (AHB1PERIPH_BASEADDR + 0x0800)
- #define GPIOD_BASEADDR (AHB1PERIPH_BASEADDR + 0x0C00)
- #define GPIOE_BASEADDR (AHB1PERIPH_BASEADDR + 0x1000)
- #define GPIOF_BASEADDR (AHB1PERIPH_BASEADDR + 0x1400)
- #define GPIOG_BASEADDR (AHB1PERIPH_BASEADDR + 0x1800)
- #define GPIOH_BASEADDR (AHB1PERIPH_BASEADDR + 0x1C00)
- #define GPIOI_BASEADDR (AHB1PERIPH_BASEADDR + 0x2000)
- #define RCC_BASEADDR (AHB1PERIPH_BASEADDR + 0x3800)
- #define DMA1_BASEADDR (AHB1PERIPH_BASEADDR + 0x6000)
- #define DMA2_BASEADDR (AHB1PERIPH_BASEADDR + 0x6400)
- #define CRC_BASEADDR (AHB1PERIPH_BASEADDR + 0x3000)
- #define FIR_BASEADDR (AHB1PERIPH_BASEADDR + 0x3C00)
- #define I2C1_BASEADDR (APB1PERIPH_BASEADDR+0x5400)
- #define I2C2_BASEADDR (APB1PERIPH_BASEADDR+0x5800)
- #define I2C3_BASEADDR (APB1PERIPH_BASEADDR+0x5C00)
- #define SPI2_BASEADDR (APB1PERIPH_BASEADDR+0x3800)
- #define SPI3_BASEADDR (APB1PERIPH_BASEADDR+0x3C00)
- #define USART2_BASEADDR (APB1PERIPH_BASEADDR+0x4400)
- #define USART3_BASEADDR (APB1PERIPH_BASEADDR+0x4800)
- #define UART4_BASEADDR (APB1PERIPH_BASEADDR+0x4C00)
- #define UART5_BASEADDR (APB1PERIPH_BASEADDR+0x5000)

4.23.1 Detailed Description

Base addresses of peripherals connected to the APB1 bus.

4.23.2 Macro Definition Documentation

4.23.2.1 CRC_BASEADDR

```
#define CRC_BASEADDR (AHB1PERIPH_BASEADDR + 0x3000)
```

Base address of CRC peripheral

4.23.2.2 DMA1_BASEADDR

```
#define DMA1_BASEADDR (AHB1PERIPH_BASEADDR + 0x6000)
```

Base address of DMA1 peripheral

4.23.2.3 DMA2_BASEADDR

```
#define DMA2_BASEADDR (AHB1PERIPH_BASEADDR + 0x6400)
```

Base address of DMA2 peripheral

4.23.2.4 FIR_BASEADDR

```
#define FIR_BASEADDR (AHB1PERIPH_BASEADDR + 0x3C00)
```

Base address of Flash Interface peripheral

4.23.2.5 GPIOA_BASEADDR

```
#define GPIOA_BASEADDR (AHB1PERIPH_BASEADDR + 0x0000)
```

Base address of GPIOA peripheral

4.23.2.6 GPIOB_BASEADDR

```
#define GPIOB_BASEADDR (AHB1PERIPH_BASEADDR + 0x0400)
```

Base address of GPIOB peripheral

4.23.2.7 GPIOC_BASEADDR

```
#define GPIOC_BASEADDR (AHB1PERIPH_BASEADDR + 0x0800)
```

Base address of GPIOC peripheral

4.23.2.8 GPIOD_BASEADDR

```
#define GPIOD_BASEADDR (AHB1PERIPH_BASEADDR + 0x0C00)
```

Base address of GPIOD peripheral

4.23.2.9 GPIOE_BASEADDR

```
#define GPIOE_BASEADDR (AHB1PERIPH_BASEADDR + 0x1000)
```

Base address of GPIOE peripheral

4.23.2.10 GPIOF_BASEADDR

```
#define GPIOF_BASEADDR (AHB1PERIPH_BASEADDR + 0x1400)
```

Base address of GPIOF peripheral

4.23.2.11 GPIOG_BASEADDR

```
#define GPIOG_BASEADDR (AHB1PERIPH_BASEADDR + 0x1800)
```

Base address of GPIOG peripheral

4.23.2.12 GPIOH_BASEADDR

```
#define GPIOH_BASEADDR (AHB1PERIPH_BASEADDR + 0x1C00)
```

Base address of GPIOH peripheral

4.23.2.13 GPIOI_BASEADDR

```
#define GPIOI_BASEADDR (AHB1PERIPH_BASEADDR + 0x2000)
```

Base address of GPIOI peripheral

4.23.2.14 I2C1_BASEADDR

```
#define I2C1_BASEADDR (APB1PERIPH_BASEADDR+0x5400)
```

Base address of I2C1 peripheral

4.23.2.15 I2C2_BASEADDR

```
#define I2C2_BASEADDR (APB1PERIPH_BASEADDR+0x5800)
```

Base address of I2C2 peripheral

4.23.2.16 I2C3_BASEADDR

```
#define I2C3_BASEADDR (APB1PERIPH_BASEADDR+0x5C00)
```

Base address of I2C3 peripheral

4.23.2.17 RCC_BASEADDR

```
#define RCC_BASEADDR (AHB1PERIPH_BASEADDR + 0x3800)
```

Base address of RCC peripheral

4.23.2.18 SPI2_BASEADDR

```
#define SPI2_BASEADDR (APB1PERIPH_BASEADDR+0x3800)
```

Base address of SPI2 peripheral

4.23.2.19 SPI3_BASEADDR

```
#define SPI3_BASEADDR (APB1PERIPH_BASEADDR+0x3C00)
```

Base address of SPI3 peripheral

4.23.2.20 UART4_BASEADDR

```
#define UART4_BASEADDR (APB1PERIPH_BASEADDR+0x4C00)
```

Base address of UART4 peripheral

4.23.2.21 UART5_BASEADDR

```
#define UART5_BASEADDR (APB1PERIPH_BASEADDR+0x5000)
```

Base address of UART5 peripheral

4.23.2.22 USART2_BASEADDR

```
#define USART2_BASEADDR (APB1PERIPH_BASEADDR+0x4400)
```

Base address of USART2 peripheral

4.23.2.23 USART3_BASEADDR

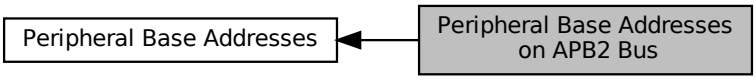
```
#define USART3_BASEADDR (APB1PERIPH_BASEADDR+0x4800)
```

Base address of USART3 peripheral

4.24 Peripheral Base Addresses on APB2 Bus

Base addresses of peripherals connected to the APB2 bus.

Collaboration diagram for Peripheral Base Addresses on APB2 Bus:



Macros

- #define SPI1_BASEADDR (APB2PERIPH_BASEADDR+0x3000)
- #define SPI4_BASEADDR (APB2PERIPH_BASEADDR+0x3400)
- #define USART1_BASEADDR (APB2PERIPH_BASEADDR+0x1000)
- #define USART6_BASEADDR (APB2PERIPH_BASEADDR+0x1400)
- #define SYSCFG_BASEADDR (APB2PERIPH_BASEADDR+0x3800)
- #define EXTI_BASEADDR (APB2PERIPH_BASEADDR+0x3C00)

4.24.1 Detailed Description

Base addresses of peripherals connected to the APB2 bus.

4.24.2 Macro Definition Documentation

4.24.2.1 EXTI_BASEADDR

```
#define EXTI_BASEADDR (APB2PERIPH_BASEADDR+0x3C00)
```

Base address of EXTI peripheral

4.24.2.2 SPI1_BASEADDR

```
#define SPI1_BASEADDR (APB2PERIPH_BASEADDR+0x3000)
```

Base address of SPI1 peripheral

4.24.2.3 SPI4_BASEADDR

```
#define SPI4_BASEADDR (APB2PERIPH_BASEADDR+0x3400)
```

Base address of SPI4 peripheral

4.24.2.4 SYSCFG_BASEADDR

```
#define SYSCFG_BASEADDR (APB2PERIPH_BASEADDR+0x3800)
```

Base address of SYSCFG peripheral

4.24.2.5 USART1_BASEADDR

```
#define USART1_BASEADDR (APB2PERIPH_BASEADDR+0x1000)
```

Base address of USART1 peripheral

4.24.2.6 USART6_BASEADDR

```
#define USART6_BASEADDR (APB2PERIPH_BASEADDR+0x1400)
```

Base address of USART6 peripheral

4.25 Peripheral Register Definitions

Structures defining the register layouts for various peripherals.

Collaboration diagram for Peripheral Register Definitions:

Classes

- struct GPIO_RegDef_t

  GPIO peripheral register definition structure.
- struct RCC_RegDef_t

  RCC peripheral register definition structure.
- struct EXTI_RegDef_t

  EXTI peripheral register definition structure.
- struct SYSCFG_RegDef_t

  SYSCFG peripheral register definition structure.
- struct SPI_RegDef_t

  SPI peripheral register definition structure.
- struct I2C_RegDef_t

  I2C peripheral register definition structure.
- struct USART_RegDef_t

  USART peripheral register definition structure.

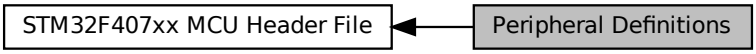
4.25.1 Detailed Description

Structures defining the register layouts for various peripherals.

4.26 Peripheral Definitions

Definitions for various peripheral instances based on their base addresses.

Collaboration diagram for Peripheral Definitions:



Macros

- #define **GPIOA** ((GPIO_RegDef_t*)GPIOA_BASEADDR)
- #define **GPIOB** ((GPIO_RegDef_t*)GPIOB_BASEADDR)
- #define **GPIOC** ((GPIO_RegDef_t*)GPIOC_BASEADDR)
- #define **GPIOD** ((GPIO_RegDef_t*)GPIOD_BASEADDR)
- #define **GPIOE** ((GPIO_RegDef_t*)GPIOE_BASEADDR)
- #define **GPIOF** ((GPIO_RegDef_t*)GPIOF_BASEADDR)
- #define **GPIOG** ((GPIO_RegDef_t*)GPIOG_BASEADDR)
- #define **GPIOH** ((GPIO_RegDef_t*)GPIOH_BASEADDR)
- #define **GPIOI** ((GPIO_RegDef_t*)GPIOI_BASEADDR)
- #define **RCC** ((RCC_RegDef_t*)RCC_BASEADDR)
- #define **EXTI** ((EXTI_RegDef_t*)EXTI_BASEADDR)

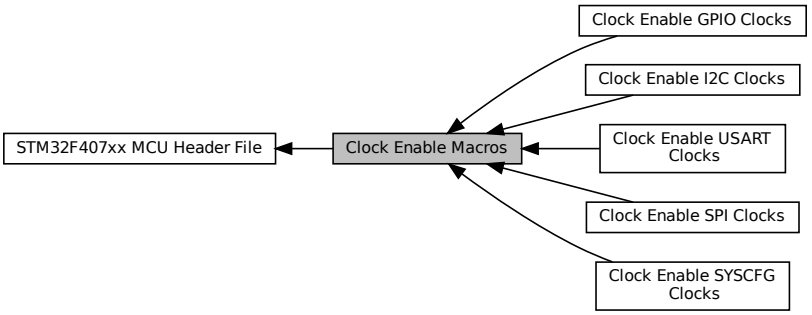
- #define **SYSCFG** ((SYSCFG_RegDef_t*)SYSCFG_BASEADDR)
- #define **SPI1** ((SPI_RegDef_t*)SPI1_BASEADDR)
- #define **SPI2** ((SPI_RegDef_t*)SPI2_BASEADDR)
- #define **SPI3** ((SPI_RegDef_t*)SPI3_BASEADDR)
- #define **SPI4** ((SPI_RegDef_t*)SPI4_BASEADDR)
- #define **I2C1** ((I2C_RegDef_t*)I2C1_BASEADDR)
- #define **I2C2** ((I2C_RegDef_t*)I2C2_BASEADDR)
- #define **I2C3** ((I2C_RegDef_t*)I2C3_BASEADDR)
- #define **USART1** ((USART_RegDef_t*)USART1_BASEADDR)
- #define **USART2** ((USART_RegDef_t*)USART2_BASEADDR)
- #define **USART3** ((USART_RegDef_t*)USART3_BASEADDR)
- #define **UART4** ((USART_RegDef_t*)UART4_BASEADDR)
- #define **UART5** ((USART_RegDef_t*)UART5_BASEADDR)
- #define **USART6** ((USART_RegDef_t*)USART6_BASEADDR)

4.26.1 Detailed Description

Definitions for various peripheral instances based on their base addresses.

4.27 Clock Enable Macros

Collaboration diagram for Clock Enable Macros:



Modules

- Clock Enable GPIO Clocks

  Macros for enabling clocks to GPIO peripherals.
- Clock Enable I2C Clocks

  Macros for enabling clocks to I2C peripherals.
- Clock Enable SPI Clocks

  Macros for enabling clocks to SPI peripherals.
- Clock Enable USART Clocks

  Macros for enabling clocks to USART peripherals.
- Clock Enable SYSCFG Clocks

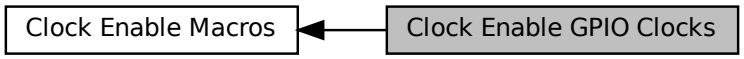
  Macros for enabling clocks to SYSCFG peripherals.

4.27.1 Detailed Description

4.28 Clock Enable GPIO Clocks

Macros for enabling clocks to GPIO peripherals.

Collaboration diagram for Clock Enable GPIO Clocks:



Macros

- #define **GPIOA_PCLK_EN**() (RCC->AHB1ENR |= (1<<0))
- #define **GPIOB_PCLK_EN**() (RCC->AHB1ENR |= (1<<1))
- #define **GPIOC_PCLK_EN**() (RCC->AHB1ENR |= (1<<2))
- #define **GPIOD_PCLK_EN**() (RCC->AHB1ENR |= (1<<3))
- #define **GPIOE_PCLK_EN**() (RCC->AHB1ENR |= (1<<4))
- #define **GPIOF_PCLK_EN**() (RCC->AHB1ENR |= (1<<5))
- #define **GPIOG_PCLK_EN**() (RCC->AHB1ENR |= (1<<6))
- #define **GPIOH_PCLK_EN**() (RCC->AHB1ENR |= (1<<7))
- #define **GPIOI_PCLK_EN**() (RCC->AHB1ENR |= (1<<8))

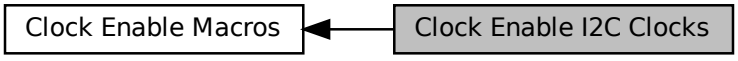
4.28.1 Detailed Description

Macros for enabling clocks to GPIO peripherals.

4.29 Clock Enable I2C Clocks

Macros for enabling clocks to I2C peripherals.

Collaboration diagram for Clock Enable I2C Clocks:



Macros

- #define **I2C1_PCLK_EN**() (RCC->APB1ENR |= (1<<21))
- #define **I2C2_PCLK_EN**() (RCC->APB1ENR |= (1<<22))
- #define **I2C3_PCLK_EN**() (RCC->APB1ENR |= (1<<23))

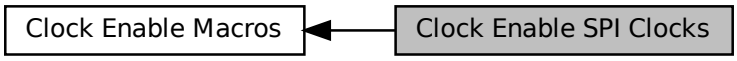
4.29.1 Detailed Description

Macros for enabling clocks to I2C peripherals.

4.30 Clock Enable SPI Clocks

Macros for enabling clocks to SPI peripherals.

Collaboration diagram for Clock Enable SPI Clocks:



Macros

- #define **SPI1_PCLK_EN**() (RCC->APB2ENR |= (1<<12))
- #define **SPI2_PCLK_EN**() (RCC->APB1ENR |= (1<<14))
- #define **SPI3_PCLK_EN**() (RCC->APB1ENR |= (1<<15))
- #define **SPI4_PCLK_EN**() (RCC->APB2ENR |= (1<<13))

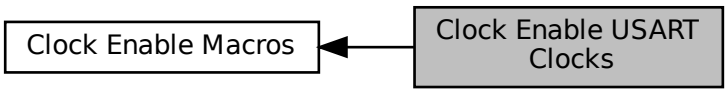
4.30.1 Detailed Description

Macros for enabling clocks to SPI peripherals.

4.31 Clock Enable USART Clocks

Macros for enabling clocks to USART peripherals.

Collaboration diagram for Clock Enable USART Clocks:



Macros

- #define **USART1_PCLK_EN**() (RCC->APB2ENR |= (1<<4))
- #define **USART2_PCLK_EN**() (RCC->APB1ENR |= (1<<17))
- #define **USART3_PCLK_EN**() (RCC->APB1ENR |= (1<<18))
- #define **UART4_PCLK_EN**() (RCC->APB1ENR |= (1<<19))
- #define **UART5_PCLK_EN**() (RCC->APB1ENR |= (1<<20))
- #define **USART6_PCLK_EN**() (RCC->APB2ENR |= (1<<5))

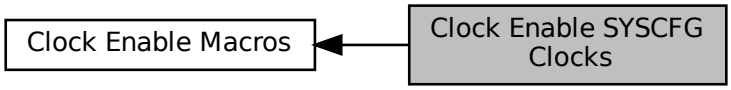
4.31.1 Detailed Description

Macros for enabling clocks to USART peripherals.

4.32 Clock Enable SYSCFG Clocks

Macros for enabling clocks to SYSCFG peripherals.

Collaboration diagram for Clock Enable SYSCFG Clocks:



Macros

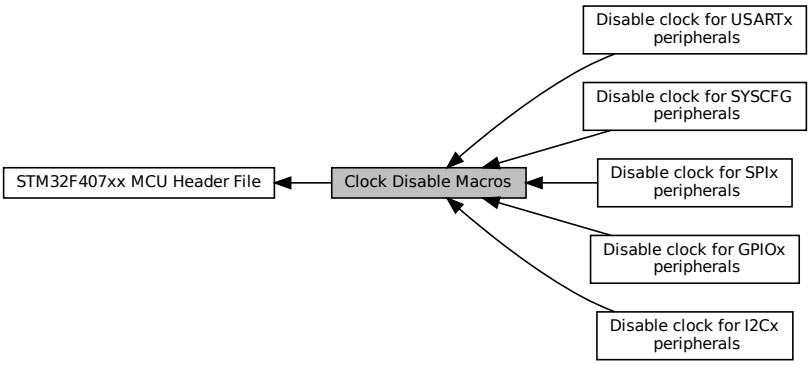
- #define **SYSCFG_PCLK_EN**() (RCC->APB2ENR |= (1<<14))

4.32.1 Detailed Description

Macros for enabling clocks to SYSCFG peripherals.

4.33 Clock Disable Macros

Collaboration diagram for Clock Disable Macros:



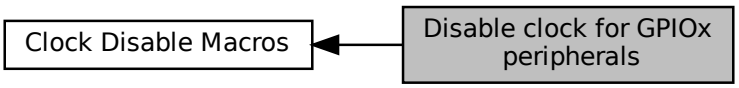
Modules

- Disable clock for GPIOx peripherals
- Disable clock for I2Cx peripherals
- Disable clock for SPIx peripherals
- Disable clock for USARTx peripherals
- Disable clock for SYSCFG peripherals

4.33.1 Detailed Description

4.34 Disable clock for GPIOx peripherals

Collaboration diagram for Disable clock for GPIOx peripherals:



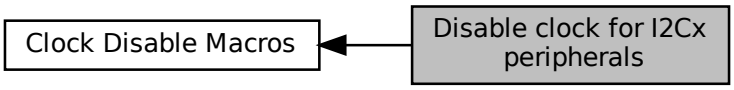
Macros

- #define **GPIOA_PCLK_DI**() (RCC->AHB1ENR &= ~(uint32_t)(1<<0))
- #define **GPIOB_PCLK_DI**() (RCC->AHB1ENR &= ~(uint32_t)(1<<1))
- #define **GPIOC_PCLK_DI**() (RCC->AHB1ENR &= ~(uint32_t)(1<<2))
- #define **GPIOD_PCLK_DI**() (RCC->AHB1ENR &= ~(uint32_t)(1<<3))
- #define **GPIOE_PCLK_DI**() (RCC->AHB1ENR &= ~(uint32_t)(1<<4))
- #define **GPIOF_PCLK_DI**() (RCC->AHB1ENR &= ~(uint32_t)(1<<5))
- #define **GPIOG_PCLK_DI**() (RCC->AHB1ENR &= ~(uint32_t)(1<<6))
- #define **GPIOH_PCLK_DI**() (RCC->AHB1ENR &= ~(uint32_t)(1<<7))
- #define **GPIOI_PCLK_DI**() (RCC->AHB1ENR &= ~(uint32_t)(1<<8))

4.34.1 Detailed Description

4.35 Disable clock for I2Cx peripherals

Collaboration diagram for Disable clock for I2Cx peripherals:



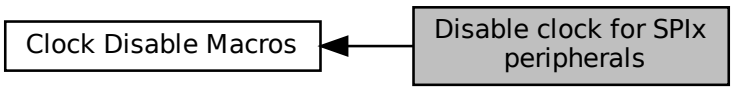
Macros

- #define **I2C1_PCLK_DI**() (RCC->APB1ENR &= ~(uint32_t)(1<<21))
- #define **I2C2_PCLK_DI**() (RCC->APB1ENR &= ~(uint32_t)(1<<22))
- #define **I2C3_PCLK_DI**() (RCC->APB1ENR &= ~(uint32_t)(1<<23))

4.35.1 Detailed Description

4.36 Disable clock for SPIx peripherals

Collaboration diagram for Disable clock for SPIx peripherals:



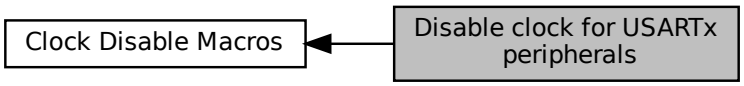
Macros

- #define **SPI1_PCLK_DI**() (RCC->APB2ENR &= ~(uint32_t)(1<<12))
- #define **SPI2_PCLK_DI**() (RCC->APB1ENR &= ~(uint32_t)(1<<14))
- #define **SPI3_PCLK_DI**() (RCC->APB1ENR &= ~(uint32_t)(1<<15))
- #define **SPI4_PCLK_DI**() (RCC->APB2ENR &= ~(uint32_t)(1<<13))

4.36.1 Detailed Description

4.37 Disable clock for USARTx peripherals

Collaboration diagram for Disable clock for USARTx peripherals:



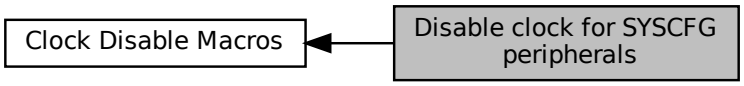
Macros

- #define **USART1_PCLK_DI**() (RCC->APB2ENR &= ~(uint32_t)(1<<4))
- #define **USART2_PCLK_DI**() (RCC->APB1ENR &= ~(uint32_t)(1<<17))
- #define **USART3_PCLK_DI**() (RCC->APB1ENR &= ~(uint32_t)(1<<18))
- #define **UART4_PCLK_DI**() (RCC->APB1ENR &= ~(uint32_t)(1<<19))
- #define **UART5_PCLK_DI**() (RCC->APB1ENR &= ~(uint32_t)(1<<20))
- #define **USART6_PCLK_DI**() (RCC->APB2ENR &= ~(uint32_t)(1<<5))

4.37.1 Detailed Description

4.38 Disable clock for SYSCFG peripherals

Collaboration diagram for Disable clock for SYSCFG peripherals:



Macros

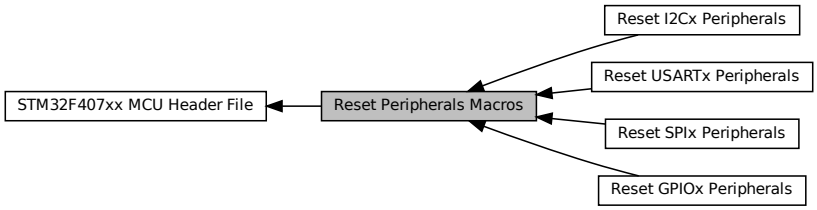
- #define **SYSCFG_PCLK_DI**() (RCC->APB2ENR &= ~(uint32_t)(1<<14))

4.38.1 Detailed Description

4.39 Reset Peripherals Macros

Macros for resetting various peripherals.

Collaboration diagram for Reset Peripherals Macros:



Modules

- Reset GPIOx Peripherals

  Macros for resetting GPIO peripherals.
- Reset I2Cx Peripherals

  Macros for resetting I2C peripherals.
- Reset SPIx Peripherals

  Macros for resetting SPI peripherals.
- Reset USARTx Peripherals

  Macros for resetting USART peripherals.

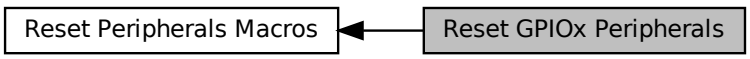
4.39.1 Detailed Description

Macros for resetting various peripherals.

4.40 Reset GPIOx Peripherals

Macros for resetting GPIO peripherals.

Collaboration diagram for Reset GPIOx Peripherals:



Macros

- #define **GPIOA_REG_RESET**() do{ (RCC->AHB1RSTR |= (1<<0)); (RCC->AHB1RSTR &= ~(1<<0));}while(0)
- #define **GPIOB_REG_RESET**() do{ (RCC->AHB1RSTR |= (1<<1)); (RCC->AHB1RSTR &= ~(1<<1));}while(0)
- #define **GPIOC_REG_RESET**() do{ (RCC->AHB1RSTR |= (1<<2)); (RCC->AHB1RSTR &= ~(1<<2));}while(0)
- #define **GPIOD_REG_RESET**() do{ (RCC->AHB1RSTR |= (1<<3)); (RCC->AHB1RSTR &= ~(1<<3));}while(0)
- #define **GPIOE_REG_RESET**() do{ (RCC->AHB1RSTR |= (1<<4)); (RCC->AHB1RSTR &= ~(1<<4));}while(0)
- #define **GPIOF_REG_RESET**() do{ (RCC->AHB1RSTR |= (1<<5)); (RCC->AHB1RSTR &= ~(1<<5));}while(0)
- #define **GPIOG_REG_RESET**() do{ (RCC->AHB1RSTR |= (1<<6)); (RCC->AHB1RSTR &= ~(1<<6));}while(0)
- #define **GPIOH_REG_RESET**() do{ (RCC->AHB1RSTR |= (1<<7)); (RCC->AHB1RSTR &= ~(1<<7));}while(0)
- #define **GPIOI_REG_RESET**() do{ (RCC->AHB1RSTR |= (1<<8)); (RCC->AHB1RSTR &= ~(1<<8));}while(0)

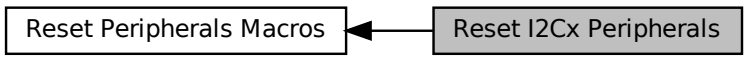
4.40.1 Detailed Description

Macros for resetting GPIO peripherals.

4.41 Reset I2Cx Peripherals

Macros for resetting I2C peripherals.

Collaboration diagram for Reset I2Cx Peripherals:



Macros

- #define **I2C1_REG_RESET**() do{ (RCC->APB1RSTR |= (1<<21)); (RCC->APB1RSTR &= ~(1<<21));}while(0)
- #define **I2C2_REG_RESET**() do{ (RCC->APB1RSTR |= (1<<22)); (RCC->APB1RSTR &= ~(1<<22));}while(0)
- #define **I2C3_REG_RESET**() do{ (RCC->APB1RSTR |= (1<<23)); (RCC->APB1RSTR &= ~(1<<23));}while(0)

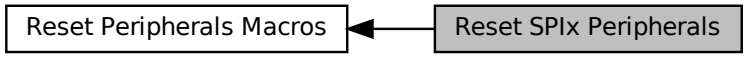
4.41.1 Detailed Description

Macros for resetting I2C peripherals.

4.42 Reset SPIx Peripherals

Macros for resetting SPI peripherals.

Collaboration diagram for Reset SPIx Peripherals:



Macros

- #define **SPI1_REG_RESET**() do{ (RCC->APB2RSTR |= (1<<12)); (RCC->APB2RSTR &= ~(1<<12));}while(0)
- #define **SPI2_REG_RESET**() do{ (RCC->APB1RSTR |= (1<<14)); (RCC->APB1RSTR &= ~(1<<14));}while(0)
- #define **SPI3_REG_RESET**() do{ (RCC->APB1RSTR |= (1<<15)); (RCC->APB1RSTR &= ~(1<<15));}while(0)
- #define **SPI4_REG_RESET**() do{ (RCC->APB2RSTR |= (1<<13)); (RCC->APB2RSTR &= ~(1<<13));}while(0)

4.42.1 Detailed Description

Macros for resetting SPI peripherals.

4.43 Reset USARTx Peripherals

Macros for resetting USART peripherals.

Collaboration diagram for Reset USARTx Peripherals:

Macros

- #define **USART1_REG_RESET**() do{ (RCC->APB2RSTR |= (1<<4)); (RCC->APB2RSTR &= ~(1<<4));}while(0)
- #define **USART2_REG_RESET**() do{ (RCC->APB1RSTR |= (1<<17)); (RCC->APB1RSTR &= ~(1<<17));}while(0)
- #define **USART3_REG_RESET**() do{ (RCC->APB1RSTR |= (1<<18)); (RCC->APB1RSTR &= ~(1<<18));}while(0)
- #define **UART4_REG_RESET**() do{ (RCC->APB1RSTR |= (1<<19)); (RCC->APB1RSTR &= ~(1<<19));}while(0)
- #define **UART5_REG_RESET**() do{ (RCC->APB1RSTR |= (1<<20)); (RCC->APB1RSTR &= ~(1<<20));}while(0)
- #define **USART6_REG_RESET**() do{ (RCC->APB2RSTR |= (1<<5)); (RCC->APB2RSTR &= ~(1<<5));}while(0)

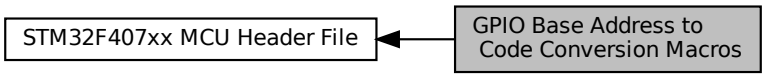
4.43.1 Detailed Description

Macros for resetting USART peripherals.

4.44 GPIO Base Address to Code Conversion Macros

Macros for converting GPIO base addresses to corresponding port codes.

Collaboration diagram for GPIO Base Address to Code Conversion Macros:



Macros

- #define GPIO_BASEADDR_TO_CODE(x)

  Macro to convert GPIO base address to port code.

4.44.1 Detailed Description

Macros for converting GPIO base addresses to corresponding port codes.

4.44.2 Macro Definition Documentation

4.44.2.1 GPIO_BASEADDR_TO_CODE

```
#define GPIO_BASEADDR_TO_CODE(
              x )
```

Value:

```
(   (x == GPIOA)?0:\
(x == GPIOB)?1:\
(x == GPIOC)?2:\
(x == GPIOD)?3:\
(x == GPIOE)?4:\
(x == GPIOF)?5:\
(x == GPIOG)?6:\
(x == GPIOH)?7:\
(x == GPIOI)?8:0 )
```

Macro to convert GPIO base address to port code.

Parameters

x	GPIOx base address.

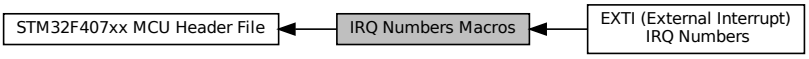
Returns

Corresponding port code for the given GPIO base address.

4.45 IRQ Numbers Macros

Macros for interrupt request numbers and priority levels.

Collaboration diagram for IRQ Numbers Macros:



Modules

- EXTI (External Interrupt) IRQ Numbers

  IRQ numbers for EXTI (External Interrupt) sources.

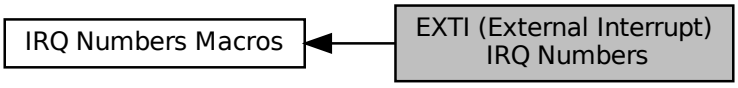
4.45.1 Detailed Description

Macros for interrupt request numbers and priority levels.

4.46 EXTI (External Interrupt) IRQ Numbers

IRQ numbers for EXTI (External Interrupt) sources.

Collaboration diagram for EXTI (External Interrupt) IRQ Numbers:



Macros

- #define **IRQ_NO_EXTI0** 6
- #define **IRQ_NO_EXTI1** 7
- #define **IRQ_NO_EXTI2** 8
- #define **IRQ_NO_EXTI3** 9
- #define **IRQ_NO_EXTI4** 10
- #define **IRQ_NO_EXTI9_5** 23
- #define **IRQ_NO_EXTI15_10** 40
- #define **IRQ_NO_SPI1** 35
- #define **IRQ_NO_SPI2** 36
- #define **IRQ_NO_SPI3** 51
- #define **IRQ_NO_SPI4** 84
- #define **IRQ_NO_I2C1_EV** 31
- #define **IRQ_NO_I2C1_ER** 32
- #define **IRQ_NO_I2C2_EV** 33
- #define **IRQ_NO_I2C2_ER** 34
- #define **IRQ_NO_I2C3_EV** 79
- #define **IRQ_NO_I2C3_ER** 80
- #define **IRQ_NO_USART1** 37
- #define **IRQ_NO_USART2** 38
- #define **IRQ_NO_USART3** 39
- #define **IRQ_NO_UART4** 52
- #define **IRQ_NO_UART5** 53
- #define **IRQ_NO_USART6** 71

4.46.1 Detailed Description

IRQ numbers for EXTI (External Interrupt) sources.

IRQ numbers for UART sources.

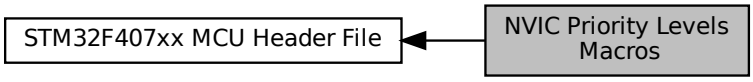
IRQ numbers for I2C sources.

IRQ numbers for SPI sources.

4.47 NVIC Priority Levels Macros

Macros for all possible priority levels for NVIC.

Collaboration diagram for NVIC Priority Levels Macros:



Macros

- #define **NVIC_IRQ_PRI0** 0
- #define **NVIC_IRQ_PRI1** 1
- #define **NVIC_IRQ_PRI2** 2
- #define **NVIC_IRQ_PRI3** 3
- #define **NVIC_IRQ_PRI4** 4
- #define **NVIC_IRQ_PRI5** 5
- #define **NVIC_IRQ_PRI6** 6
- #define **NVIC_IRQ_PRI7** 7
- #define **NVIC_IRQ_PRI8** 8
- #define **NVIC_IRQ_PRI9** 9
- #define **NVIC_IRQ_PRI10** 10
- #define **NVIC_IRQ_PRI11** 11
- #define **NVIC_IRQ_PRI12** 12
- #define **NVIC_IRQ_PRI13** 13
- #define **NVIC_IRQ_PRI14** 14
- #define **NVIC_IRQ_PRI15** 15

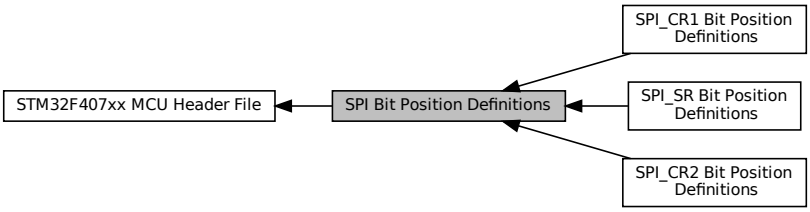
4.47.1 Detailed Description

Macros for all possible priority levels for NVIC.

4.48 SPI Bit Position Definitions

Bit position definitions for various registers in the SPI peripheral.

Collaboration diagram for SPI Bit Position Definitions:



Modules

- SPI_CR1 Bit Position Definitions

  Bit position definitions for SPI_CR1 register.
- SPI_CR2 Bit Position Definitions

  Bit position definitions for SPI_CR2 register.
- SPI_SR Bit Position Definitions

  Bit position definitions for SPI_SR register.

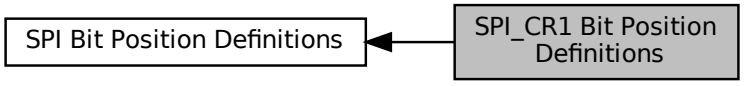
4.48.1 Detailed Description

Bit position definitions for various registers in the SPI peripheral.

4.49 SPI_CR1 Bit Position Definitions

Bit position definitions for SPI_CR1 register.

Collaboration diagram for SPI_CR1 Bit Position Definitions:



Macros

- #define SPI_CR1_CPHA 0
- #define SPI_CR1_CPOL 1
- #define SPI_CR1_MSTR 2
- #define SPI_CR1_BR 3
- #define SPI_CR1_SPE 6
- #define SPI_CR1_LSBFIRST 7
- #define SPI_CR1_SSI 8
- #define SPI_CR1_SSM 9
- #define SPI_CR1_RXONLY 10
- #define SPI_CR1_DFF 11
- #define SPI_CR1_CRCNEXT 12
- #define SPI_CR1_CRCEN 13
- #define SPI_CR1_BIDIOE 14
- #define SPI_CR1_BIDIMODE 15

4.49.1 Detailed Description

Bit position definitions for SPI_CR1 register.

4.49.2 Macro Definition Documentation

4.49.2.1 SPI_CR1_BIDIMODE

```
#define SPI_CR1_BIDIMODE 15
```

Bidirectional Data Mode Enable

4.49.2.2 SPI_CR1_BIDIOE

```
#define SPI_CR1_BIDIOE 14
```

Output Enable in Bidirectional Mode

4.49.2.3 SPI_CR1_BR

```
#define SPI_CR1_BR 3
```

Baud Rate Control

4.49.2.4 SPI_CR1_CPHA

```
#define SPI_CR1_CPHA 0
```

Clock Phase

4.49.2.5 SPI_CR1_CPOL

```
#define SPI_CR1_CPOL 1
```

Clock Polarity

4.49.2.6 SPI_CR1_CRCEN

```
#define SPI_CR1_CRCEN 13
```

CRC Calculation Enable

4.49.2.7 SPI_CR1_CRCNEXT

```
#define SPI_CR1_CRCNEXT 12
```

CRC Transfer Next

4.49.2.8 SPI_CR1_DFF

```
#define SPI_CR1_DFF 11
```

Data Frame Format

4.49.2.9 SPI_CR1_LSBFIRST

```
#define SPI_CR1_LSBFIRST 7
```

Frame Format

4.49.2.10 SPI_CR1_MSTR

```
#define SPI_CR1_MSTR 2
```

Master Selection

4.49.2.11 SPI_CR1_RXONLY

```
#define SPI_CR1_RXONLY 10
```

Receive Only

4.49.2.12 SPI_CR1_SPE

```
#define SPI_CR1_SPE 6
```

SPI Peripheral Enable

4.49.2.13 SPI_CR1_SSI

```
#define SPI_CR1_SSI 8
```

Internal Slave Select

4.49.2.14 SPI_CR1_SSM

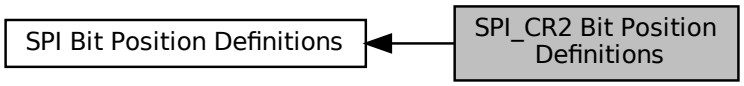
```
#define SPI_CR1_SSM 9
```

Software Slave Management

4.50 SPI_CR2 Bit Position Definitions

Bit position definitions for SPI_CR2 register.

Collaboration diagram for SPI_CR2 Bit Position Definitions:



Macros

- #define SPI_CR2_RXDMAEN 0
- #define SPI_CR2_TXDMAEN 1
- #define SPI_CR2_SSOE 2
- #define SPI_CR2_FRF 4
- #define SPI_CR2_ERRIE 5
- #define SPI_CR2_RXNEIE 6
- #define SPI_CR2_TXEIE 7

4.50.1 Detailed Description

Bit position definitions for SPI_CR2 register.

4.50.2 Macro Definition Documentation

4.50.2.1 SPI_CR2_ERRIE

```
#define SPI_CR2_ERRIE 5
```

Error Interrupt Enable

4.50.2.2 SPI_CR2_FRF

```
#define SPI_CR2_FRF 4
```

Frame Format

4.50.2.3 SPI_CR2_RXDMAEN

```
#define SPI_CR2_RXDMAEN 0
```

Rx Buffer DMA Enable

4.50.2.4 SPI_CR2_RXNEIE

```
#define SPI_CR2_RXNEIE 6
```

RX buffer Not Empty Interrupt Enable

4.50.2.5 SPI_CR2_SSOE

```
#define SPI_CR2_SSOE 2
```

SS Output Enable

4.50.2.6 SPI_CR2_TXDMAEN

```
#define SPI_CR2_TXDMAEN 1
```

Tx Buffer DMA Enable

4.50.2.7 SPI_CR2_TXEIE

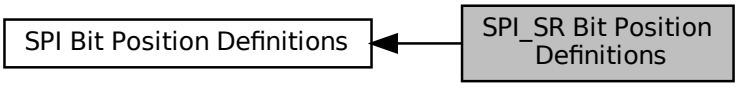
```
#define SPI_CR2_TXEIE 7
```

TX buffer Empty Interrupt Enable

4.51 SPI_SR Bit Position Definitions

Bit position definitions for SPI_SR register.

Collaboration diagram for SPI_SR Bit Position Definitions:



Macros

- #define SPI_SR_RXNE 0
- #define SPI_SR_TXE 1
- #define SPI_SR_CHSIDE 2
- #define SPI_SR_UDR 3
- #define SPI_SR_CRCERR 4
- #define SPI_SR_MODF 5
- #define SPI_SR_OVR 6
- #define SPI_SR_BSY 7
- #define SPI_SR_FRE 8

4.51.1 Detailed Description

Bit position definitions for SPI_SR register.

4.51.2 Macro Definition Documentation

4.51.2.1 SPI_SR_BSY

```
#define SPI_SR_BSY 7
```

Busy Flag

4.51.2.2 SPI_SR_CHSIDE

```
#define SPI_SR_CHSIDE 2
```

Channel Side

4.51.2.3 SPI_SR_CRCERR

```
#define SPI_SR_CRCERR 4
```

CRC Error Flag

4.51.2.4 SPI_SR_FRE

```
#define SPI_SR_FRE 8
```

Frame Format Error Flag

4.51.2.5 SPI_SR_MODF

```
#define SPI_SR_MODF 5
```

Mode Fault

4.51.2.6 SPI_SR_OVR

```
#define SPI_SR_OVR 6
```

Overrun Flag

4.51.2.7 SPI_SR_RXNE

```
#define SPI_SR_RXNE 0
```

Receive buffer Not Empty

4.51.2.8 SPI_SR_TXE

```
#define SPI_SR_TXE 1
```

Transmit buffer Empty

4.51.2.9 SPI_SR_UDR

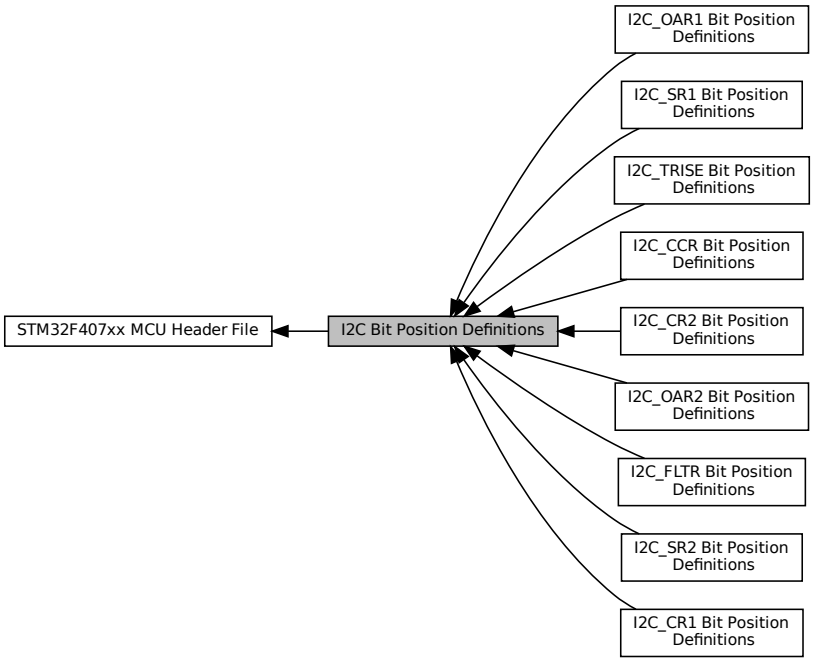
```
#define SPI_SR_UDR 3
```

Underrun Flag

4.52 I2C Bit Position Definitions

Bit position definitions for various registers in the I2C peripheral.

Collaboration diagram for I2C Bit Position Definitions:



Modules

- I2C_CR1 Bit Position Definitions

  Bit position definitions for I2C_CR1 register.
- I2C_OAR1 Bit Position Definitions

  Bit position definitions for I2C_OAR1 register.
- I2C_OAR2 Bit Position Definitions

  Bit position definitions for I2C_OAR2 register.
- I2C_CR2 Bit Position Definitions

  Bit position definitions for I2C_CR2 register.
- I2C_SR1 Bit Position Definitions

  Bit position definitions for I2C_SR1 register.
- I2C_SR2 Bit Position Definitions

  Bit position definitions for I2C_SR2 register.
- I2C_CCR Bit Position Definitions

  Bit position definitions for I2C_CCR register.
- I2C_TRISE Bit Position Definitions

  Bit position definitions for I2C_TRISE register.
- I2C_FLTR Bit Position Definitions

  Bit position definitions for I2C_FLTR register.

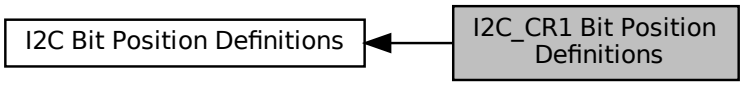
4.52.1 Detailed Description

Bit position definitions for various registers in the I2C peripheral.

4.53 I2C_CR1 Bit Position Definitions

Bit position definitions for I2C_CR1 register.

Collaboration diagram for I2C_CR1 Bit Position Definitions:



Macros

- #define I2C_CR1_PE 0
- #define I2C_CR1_SMBUS 1
- #define I2C_CR1_SMBTYPE 3
- #define I2C_CR1_ENARP 4
- #define I2C_CR1_ENPEC 5
- #define I2C_CR1_ENGC 6
- #define I2C_CR1_NOSTRETCH 7
- #define I2C_CR1_START 8
- #define I2C_CR1_STOP 9
- #define I2C_CR1_ACK 10
- #define I2C_CR1_POS 11
- #define I2C_CR1_PEC 12
- #define I2C_CR1_ALERT 13
- #define I2C_CR1_SWRST 15

4.53.1 Detailed Description

Bit position definitions for I2C_CR1 register.

4.53.2 Macro Definition Documentation

4.53.2.1 I2C_CR1_ACK

```
#define I2C_CR1_ACK 10
```

Acknowledge Enable

4.53.2.2 I2C_CR1_ALERT

```
#define I2C_CR1_ALERT 13
```

SMBus Alert

4.53.2.3 I2C_CR1_ENARP

```
#define I2C_CR1_ENARP 4
```

ARP Enable

4.53.2.4 I2C_CR1_ENGC

```
#define I2C_CR1_ENGC 6
```

General Call Enable

4.53.2.5 I2C_CR1_ENPEC

```
#define I2C_CR1_ENPEC 5
```

PEC Enable

4.53.2.6 I2C_CR1_NOSTRETCH

```
#define I2C_CR1_NOSTRETCH 7
```

Clock Stretching Disable

4.53.2.7 I2C_CR1_PE

```
#define I2C_CR1_PE 0
```

Peripheral Enable

4.53.2.8 I2C_CR1_PEC

```
#define I2C_CR1_PEC 12
```

Packet Error Checking

4.53.2.9 I2C_CR1_POS

```
#define I2C_CR1_POS 11
```

Acknowledge/Not Acknowledge

4.53.2.10 I2C_CR1_SMBTYPE

```
#define I2C_CR1_SMBTYPE 3
```

SMBus Type

4.53.2.11 I2C_CR1_SMBUS

```
#define I2C_CR1_SMBUS 1
```

SMBus Mode

4.53.2.12 I2C_CR1_START

```
#define I2C_CR1_START 8
```

Start Generation

4.53.2.13 I2C_CR1_STOP

```
#define I2C_CR1_STOP 9
```

Stop Generation

4.53.2.14 I2C_CR1_SWRST

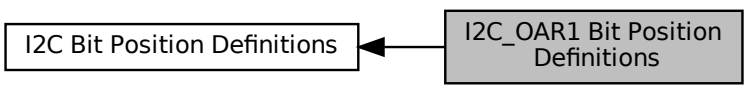
```
#define I2C_CR1_SWRST 15
```

Software Reset

4.54 I2C_OAR1 Bit Position Definitions

Bit position definitions for I2C_OAR1 register.

Collaboration diagram for I2C_OAR1 Bit Position Definitions:



Macros

- #define I2C_OAR1_ADD0 0
- #define I2C_OAR1_ADD 1
- #define I2C_OAR1_ADDMODE 15

4.54.1 Detailed Description

Bit position definitions for I2C_OAR1 register.

4.54.2 Macro Definition Documentation

4.54.2.1 I2C_OAR1_ADD

```
#define I2C_OAR1_ADD 1
```

Interface Address

4.54.2.2 I2C_OAR1_ADD0

```
#define I2C_OAR1_ADD0 0
```

Addressing Mode Bit 0

4.54.2.3 I2C_OAR1_ADDMODE

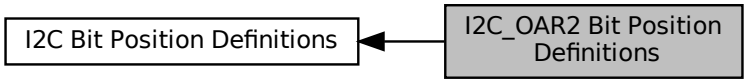
```
#define I2C_OAR1_ADDMODE 15
```

Addressing Mode

4.55 I2C_OAR2 Bit Position Definitions

Bit position definitions for I2C_OAR2 register.

Collaboration diagram for I2C_OAR2 Bit Position Definitions:



Macros

- #define I2C_OAR2_ENDUAL 0
- #define I2C_OAR2_ADD2 1

4.55.1 Detailed Description

Bit position definitions for I2C_OAR2 register.

4.55.2 Macro Definition Documentation

4.55.2.1 I2C_OAR2_ADD2

```
#define I2C_OAR2_ADD2 1
```

Interface Address 2

4.55.2.2 I2C_OAR2_ENDUAL

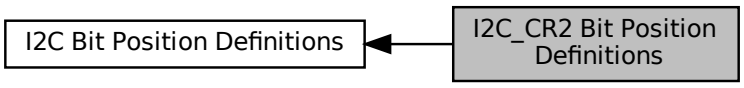
```
#define I2C_OAR2_ENDUAL 0
```

Dual Addressing Mode Enable

4.56 I2C_CR2 Bit Position Definitions

Bit position definitions for I2C_CR2 register.

Collaboration diagram for I2C_CR2 Bit Position Definitions:



Macros

- #define I2C_CR2_FREQ 0
- #define I2C_CR2_ITERREN 8
- #define I2C_CR2_ITEVTEN 9
- #define I2C_CR2_ITBUFEN 10
- #define I2C_CR2_DMAEN 11
- #define I2C_CR2_LAST 12

4.56.1 Detailed Description

Bit position definitions for I2C_CR2 register.

4.56.2 Macro Definition Documentation

4.56.2.1 I2C_CR2_DMAEN

```
#define I2C_CR2_DMAEN 11
```

DMA Requests Enable

4.56.2.2 I2C_CR2_FREQ

```
#define I2C_CR2_FREQ 0
```

Peripheral Clock Frequency

4.56.2.3 I2C_CR2_ITBUFEN

```
#define I2C_CR2_ITBUFEN 10
```

Buffer Interrupt Enable

4.56.2.4 I2C_CR2_ITERREN

```
#define I2C_CR2_ITERREN 8
```

Error Interrupt Enable

4.56.2.5 I2C_CR2_ITEVTEN

```
#define I2C_CR2_ITEVTEN 9
```

Event Interrupt Enable

4.56.2.6 I2C_CR2_LAST

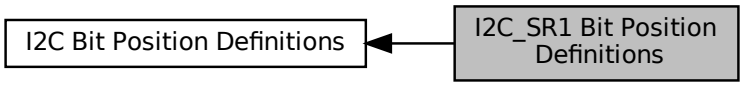
```
#define I2C_CR2_LAST 12
```

DMA Last Transfer

4.57 I2C_SR1 Bit Position Definitions

Bit position definitions for I2C_SR1 register.

Collaboration diagram for I2C_SR1 Bit Position Definitions:



Macros

- #define I2C_SR1_SB 0
- #define I2C_SR1_ADDR 1
- #define I2C_SR1_BTF 2
- #define I2C_SR1_ADD10 3
- #define I2C_SR1_STOPF 4
- #define I2C_SR1_RxNE 6
- #define I2C_SR1_TxE 7
- #define I2C_SR1_BERR 8
- #define I2C_SR1_ARLO 9
- #define I2C_SR1_AF 10
- #define I2C_SR1_OVR 11
- #define I2C_SR1_PECERR 12
- #define I2C_SR1_TIMEOUT 14
- #define I2C_SR1_SMBALERT 15

4.57.1 Detailed Description

Bit position definitions for I2C_SR1 register.

4.57.2 Macro Definition Documentation

4.57.2.1 I2C_SR1_ADD10

```
#define I2C_SR1_ADD10 3
```

10-bit Header Sent (Master mode)

4.57.2.2 I2C_SR1_ADDR

```
#define I2C_SR1_ADDR 1
```

Address Sent (master mode) / Address Matched (slave mode)

4.57.2.3 I2C_SR1_AF

```
#define I2C_SR1_AF 10
```

Acknowledge Failure

4.57.2.4 I2C_SR1_ARLO

```
#define I2C_SR1_ARLO 9
```

Arbitration Lost (master mode)

4.57.2.5 I2C_SR1_BERR

```
#define I2C_SR1_BERR 8
```

Bus Error

4.57.2.6 I2C_SR1_BTF

```
#define I2C_SR1_BTF 2
```

Byte Transfer Finished

4.57.2.7 I2C_SR1_OVR

```
#define I2C_SR1_OVR 11
```

Overrun/Underrun

4.57.2.8 I2C_SR1_PECERR

```
#define I2C_SR1_PECERR 12
```

PEC Error in reception

4.57.2.9 I2C_SR1_RxNE

```
#define I2C_SR1_RxNE 6
```

Data Register Not Empty (receivers)

4.57.2.10 I2C_SR1_SB

```
#define I2C_SR1_SB 0
```

Start Bit

4.57.2.11 I2C_SR1_SMBALERT

```
#define I2C_SR1_SMBALERT 15
```

SMBus Alert

4.57.2.12 I2C_SR1_STOPF

```
#define I2C_SR1_STOPF 4
```

Stop Detection (Slave mode)

4.57.2.13 I2C_SR1_TIMEOUT

```
#define I2C_SR1_TIMEOUT 14
```

Timeout or Tlow Error

4.57.2.14 I2C_SR1_TxE

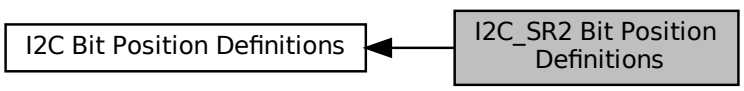
```
#define I2C_SR1_TxE 7
```

Data Register Empty (transmitters)

4.58 I2C_SR2 Bit Position Definitions

Bit position definitions for I2C_SR2 register.

Collaboration diagram for I2C_SR2 Bit Position Definitions:



Macros

- #define I2C_SR2_MSL 0
- #define I2C_SR2_BUSY 1
- #define I2C_SR2_TRA 2
- #define I2C_SR2_GENCALL 4
- #define I2C_SR2_SMBDEFAULT 5
- #define I2C_SR2_SMBHOST 6
- #define I2C_SR2_DUALF 7
- #define I2C_SR2_PEC 8

4.58.1 Detailed Description

Bit position definitions for I2C_SR2 register.

4.58.2 Macro Definition Documentation

4.58.2.1 I2C_SR2_BUSY

```
#define I2C_SR2_BUSY 1
```

Bus Busy

4.58.2.2 I2C_SR2_DUALF

```
#define I2C_SR2_DUALF 7
```

Dual Flag (Slave mode)

4.58.2.3 I2C_SR2_GENCALL

```
#define I2C_SR2_GENCALL 4
```

General Call Address (Slave mode)

4.58.2.4 I2C_SR2_MSL

```
#define I2C_SR2_MSL 0
```

Master/Slave

4.58.2.5 I2C_SR2_PEC

```
#define I2C_SR2_PEC 8
```

Packet Error Checking Register

4.58.2.6 I2C_SR2_SMBDEFAULT

```
#define I2C_SR2_SMBDEFAULT 5
```

SMBus Device Default Address (Slave mode)

4.58.2.7 I2C_SR2_SMBHOST

```
#define I2C_SR2_SMBHOST 6
```

SMBus Host Header (Slave mode)

4.58.2.8 I2C_SR2_TRA

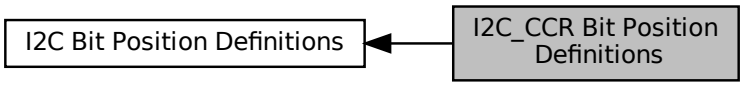
```
#define I2C_SR2_TRA 2
```

Transmitter/Receiver

4.59 I2C_CCR Bit Position Definitions

Bit position definitions for I2C_CCR register.

Collaboration diagram for I2C_CCR Bit Position Definitions:



Macros

- #define I2C_CCR_CCR 0
- #define I2C_CCR_DUTY 14
- #define I2C_CCR_FS 15

4.59.1 Detailed Description

Bit position definitions for I2C_CCR register.

4.59.2 Macro Definition Documentation

4.59.2.1 I2C_CCR_CCR

```
#define I2C_CCR_CCR 0
```

Clock Control Register in Fast/Standard mode

4.59.2.2 I2C_CCR_DUTY

```
#define I2C_CCR_DUTY 14
```

Fast Mode Duty Cycle

4.59.2.3 I2C_CCR_FS

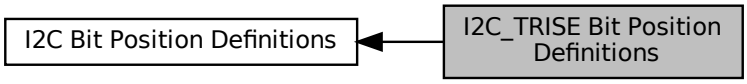
```
#define I2C_CCR_FS 15
```

I2C Master Mode Selection

4.60 I2C_TRISE Bit Position Definitions

Bit position definitions for I2C_TRISE register.

Collaboration diagram for I2C_TRISE Bit Position Definitions:



Macros

- #define I2C_TRISE_TRISE 0

4.60.1 Detailed Description

Bit position definitions for I2C_TRISE register.

4.60.2 Macro Definition Documentation

4.60.2.1 I2C_TRISE_TRISE

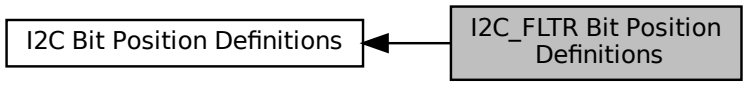
```
#define I2C_TRISE_TRISE 0
```

Maximum Rise Time in Fast/Standard mode

4.61 I2C_FLTR Bit Position Definitions

Bit position definitions for I2C_FLTR register.

Collaboration diagram for I2C_FLTR Bit Position Definitions:



Macros

- #define I2C_FLTR_DNF 0
- #define I2C_FLTR_ANOFF 4

4.61.1 Detailed Description

Bit position definitions for I2C_FLTR register.

4.61.2 Macro Definition Documentation

4.61.2.1 I2C_FLTR_ANOFF

```
#define I2C_FLTR_ANOFF 4
```

Analog Noise Filter

4.61.2.2 I2C_FLTR_DNF

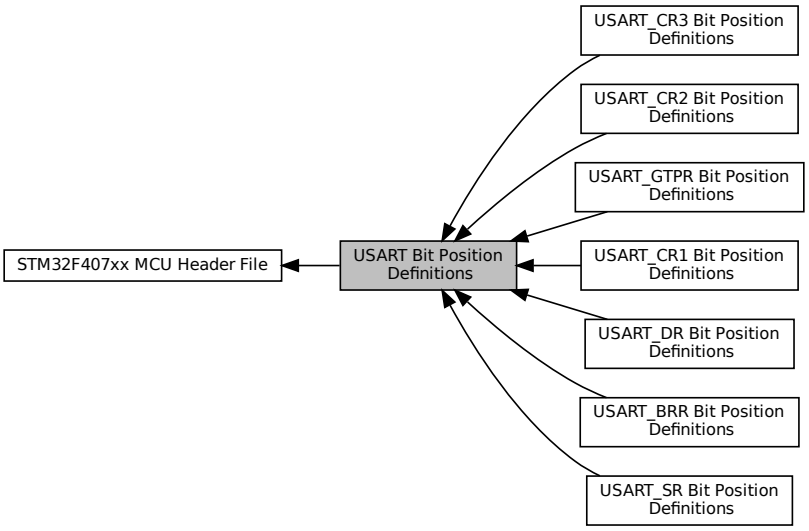
```
#define I2C_FLTR_DNF 0
```

Digital Noise Filter

4.62 USART Bit Position Definitions

Bit position definitions for various registers in the USART peripheral.

Collaboration diagram for USART Bit Position Definitions:



Modules

- USART_SR Bit Position Definitions

  Bit position definitions for USART_SR register.
- USART_DR Bit Position Definitions

  Bit position definitions for USART_DR register.
- USART_BRR Bit Position Definitions

  Bit position definitions for USART_BRR register.
- USART_CR1 Bit Position Definitions

  Bit position definitions for USART_CR1 register.
- USART_CR2 Bit Position Definitions

  Bit position definitions for USART_CR2 register.
- USART_CR3 Bit Position Definitions

  Bit position definitions for USART_CR3 register.
- USART_GTPR Bit Position Definitions

  Bit position definitions for USART_GTPR register.

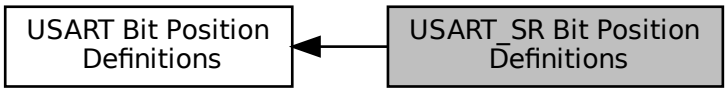
4.62.1 Detailed Description

Bit position definitions for various registers in the USART peripheral.

4.63 USART_SR Bit Position Definitions

Bit position definitions for USART_SR register.

Collaboration diagram for USART_SR Bit Position Definitions:



Macros

- #define USART_SR_PE 0
- #define USART_SR_FE 1
- #define USART_SR_NF 2
- #define USART_SR_ORE 3
- #define USART_SR_IDLE 4
- #define USART_SR_RXNE 5
- #define USART_SR_TC 6
- #define USART_SR_TXE 7
- #define USART_SR_LBD 8
- #define USART_SR_CTS 9

4.63.1 Detailed Description

Bit position definitions for USART_SR register.

4.63.2 Macro Definition Documentation

4.63.2.1 USART_SR_CTS

```
#define USART_SR_CTS 9
```

CTS Interrupt Flag

4.63.2.2 USART_SR_FE

```
#define USART_SR_FE 1
```

Framing Error Flag

4.63.2.3 USART_SR_IDLE

```
#define USART_SR_IDLE 4
```

Idle Line Detected Flag

4.63.2.4 USART_SR_LBD

```
#define USART_SR_LBD 8
```

LIN Break Detection Flag

4.63.2.5 USART_SR_NF

```
#define USART_SR_NF 2
```

Noise Flag

4.63.2.6 USART_SR_ORE

```
#define USART_SR_ORE 3
```

Overrun Error Flag

4.63.2.7 USART_SR_PE

```
#define USART_SR_PE 0
```

Parity Error Flag

4.63.2.8 USART_SR_RXNE

```
#define USART_SR_RXNE 5
```

Read Data Register Not Empty Flag

4.63.2.9 USART_SR_TC

```
#define USART_SR_TC 6
```

Transmission Complete Flag

4.63.2.10 USART_SR_TXE

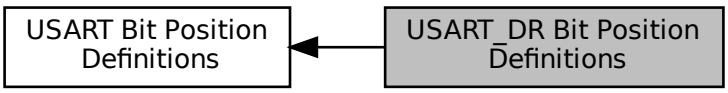
```
#define USART_SR_TXE 7
```

Transmit Data Register Empty Flag

4.64 USART_DR Bit Position Definitions

Bit position definitions for USART_DR register.

Collaboration diagram for USART_DR Bit Position Definitions:



Macros

- #define USART_DR_DR 0

4.64.1 Detailed Description

Bit position definitions for USART_DR register.

4.64.2 Macro Definition Documentation

4.64.2.1 USART_DR_DR

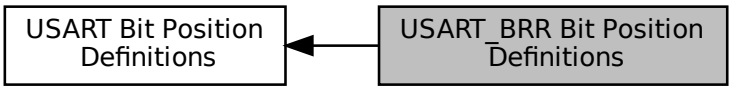
```
#define USART_DR_DR 0
```

Data Value (0-8 bits)

4.65 USART_BRR Bit Position Definitions

Bit position definitions for USART_BRR register.

Collaboration diagram for USART_BRR Bit Position Definitions:



Macros

- #define USART_BRR_DIV_FRACTION 0
- #define USART_BRR_DIV_MANTISSA 4

4.65.1 Detailed Description

Bit position definitions for USART_BRR register.

4.65.2 Macro Definition Documentation

4.65.2.1 USART_BRR_DIV_FRACTION

```
#define USART_BRR_DIV_FRACTION 0
```

Fractional part of the USARTDIV

4.65.2.2 USART_BRR_DIV_MANTISSA

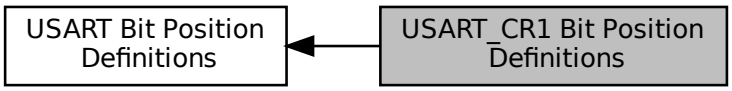
```
#define USART_BRR_DIV_MANTISSA 4
```

Mantissa part of the USARTDIV

4.66 USART_CR1 Bit Position Definitions

Bit position definitions for USART_CR1 register.

Collaboration diagram for USART_CR1 Bit Position Definitions:



Macros

- #define USART_CR1_SBK 0
- #define USART_CR1_RWU 1
- #define USART_CR1_RE 2
- #define USART_CR1_TE 3
- #define USART_CR1_IDLEIE 4
- #define USART_CR1_RXNEIE 5
- #define USART_CR1_TCIE 6
- #define USART_CR1_TXEIE 7
- #define USART_CR1_PEIE 8
- #define USART_CR1_PS 9
- #define USART_CR1_PCE 10
- #define USART_CR1_WAKE 11
- #define USART_CR1_M 12
- #define USART_CR1_UE 13
- #define USART_CR1_OVER8 15

4.66.1 Detailed Description

Bit position definitions for USART_CR1 register.

4.66.2 Macro Definition Documentation

4.66.2.1 USART_CR1_IDLEIE

```
#define USART_CR1_IDLEIE 4
```

IDLE Interrupt Enable

4.66.2.2 USART_CR1_M

```
#define USART_CR1_M 12
```

Word Length

4.66.2.3 USART_CR1_OVER8

```
#define USART_CR1_OVER8 15
```

Oversampling Mode

4.66.2.4 USART_CR1_PCE

```
#define USART_CR1_PCE 10
```

Parity Control Enable

4.66.2.5 USART_CR1_PEIE

```
#define USART_CR1_PEIE 8
```

Parity Error Interrupt Enable

4.66.2.6 USART_CR1_PS

```
#define USART_CR1_PS 9
```

Parity Selection

4.66.2.7 USART_CR1_RE

```
#define USART_CR1_RE 2
```

Receiver Enable

4.66.2.8 USART_CR1_RWU

```
#define USART_CR1_RWU 1
```

Receiver Wake-Up

4.66.2.9 USART_CR1_RXNEIE

```
#define USART_CR1_RXNEIE 5
```

RXNE Interrupt Enable

4.66.2.10 USART_CR1_SBK

```
#define USART_CR1_SBK 0
```

Send Break

4.66.2.11 USART_CR1_TCIE

```
#define USART_CR1_TCIE 6
```

Transmission Complete Interrupt Enable

4.66.2.12 USART_CR1_TE

```
#define USART_CR1_TE 3
```

Transmitter Enable

4.66.2.13 USART_CR1_TXEIE

```
#define USART_CR1_TXEIE 7
```

TXE Interrupt Enable

4.66.2.14 USART_CR1_UE

```
#define USART_CR1_UE 13
```

USART Enable

4.66.2.15 USART_CR1_WAKE

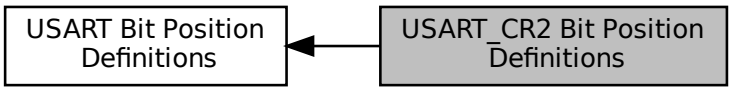
```
#define USART_CR1_WAKE 11
```

Receiver Wake-Up Method

4.67 USART_CR2 Bit Position Definitions

Bit position definitions for USART_CR2 register.

Collaboration diagram for USART_CR2 Bit Position Definitions:



Macros

- #define USART_CR2_ADD 4
- #define USART_CR2_LBDL 5
- #define USART_CR2_LBDIE 6
- #define USART_CR2_LBCL 8
- #define USART_CR2_CPHA 9
- #define USART_CR2_CPOL 10
- #define USART_CR2_CLKEN 11
- #define USART_CR2_STOP 12
- #define USART_CR2_LINEN 14

4.67.1 Detailed Description

Bit position definitions for USART_CR2 register.

4.67.2 Macro Definition Documentation

4.67.2.1 USART_CR2_ADD

```
#define USART_CR2_ADD 4
```

Address of the USART node

4.67.2.2 USART_CR2_CLKEN

```
#define USART_CR2_CLKEN 11
```

Clock Enable

4.67.2.3 USART_CR2_CPHA

```
#define USART_CR2_CPHA 9
```

Clock Phase

4.67.2.4 USART_CR2_CPOL

```
#define USART_CR2_CPOL 10
```

Clock Polarity

4.67.2.5 USART_CR2_LBCL

```
#define USART_CR2_LBCL 8
```

Last Bit Clock pulse

4.67.2.6 USART_CR2_LBDIE

```
#define USART_CR2_LBDIE 6
```

LIN Break Detection Interrupt Enable

4.67.2.7 USART_CR2_LBDL

```
#define USART_CR2_LBDL 5
```

LIN Break Detection Length

4.67.2.8 USART_CR2_LINEN

```
#define USART_CR2_LINEN 14
```

LIN mode enable

4.67.2.9 USART_CR2_STOP

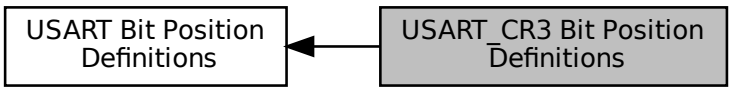
```
#define USART_CR2_STOP 12
```

STOP[1:0]: STOP bits

4.68 USART_CR3 Bit Position Definitions

Bit position definitions for USART_CR3 register.

Collaboration diagram for USART_CR3 Bit Position Definitions:



Macros

- #define USART_CR3_EIE 0
- #define USART_CR3_IREN 1
- #define USART_CR3_IRLP 2
- #define USART_CR3_HDSEL 3
- #define USART_CR3_NACK 4
- #define USART_CR3_SCEN 5
- #define USART_CR3_DMAR 6
- #define USART_CR3_DMAT 7
- #define USART_CR3_RTSE 8
- #define USART_CR3_CTSE 9
- #define USART_CR3_CTSIE 10
- #define USART_CR3_ONEBIT 11

4.68.1 Detailed Description

Bit position definitions for USART_CR3 register.

4.68.2 Macro Definition Documentation

4.68.2.1 USART_CR3_CTSE

```
#define USART_CR3_CTSE 9
```

CTS Enable

4.68.2.2 USART_CR3_CTSIE

```
#define USART_CR3_CTSIE 10
```

CTS Interrupt Enable

4.68.2.3 USART_CR3_DMAR

```
#define USART_CR3_DMAR 6
```

DMA Enable Receiver

4.68.2.4 USART_CR3_DMAT

```
#define USART_CR3_DMAT 7
```

DMA Enable Transmitter

4.68.2.5 USART_CR3_EIE

```
#define USART_CR3_EIE 0
```

Error Interrupt Enable

4.68.2.6 USART_CR3_HDSEL

```
#define USART_CR3_HDSEL 3
```

Half-Duplex Selection

4.68.2.7 USART_CR3_IREN

```
#define USART_CR3_IREN 1
```

IrDA mode Enable

4.68.2.8 USART_CR3_IRLP

```
#define USART_CR3_IRLP 2
```

IrDA Low-Power

4.68.2.9 USART_CR3_NACK

```
#define USART_CR3_NACK 4
```

Smartcard NACK Enable

4.68.2.10 USART_CR3_ONEBIT

```
#define USART_CR3_ONEBIT 11
```

One Sample Bit Method Enable

4.68.2.11 USART_CR3_RTSE

```
#define USART_CR3_RTSE 8
```

RTS Enable

4.68.2.12 USART_CR3_SCEN

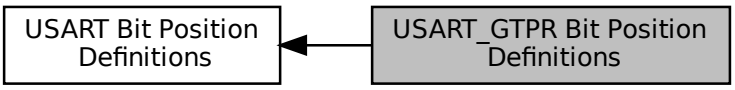
```
#define USART_CR3_SCEN 5
```

Smartcard Mode Enable

4.69 USART_GTPR Bit Position Definitions

Bit position definitions for USART_GTPR register.

Collaboration diagram for USART_GTPR Bit Position Definitions:



Macros

- #define USART_GTPR_PSC 0
- #define USART_GTPR_GT 8

4.69.1 Detailed Description

Bit position definitions for USART_GTPR register.

4.69.2 Macro Definition Documentation

4.69.2.1 USART_GTPR_GT

```
#define USART_GTPR_GT 8
```

Guard Time Value

4.69.2.2 USART_GTPR_PSC

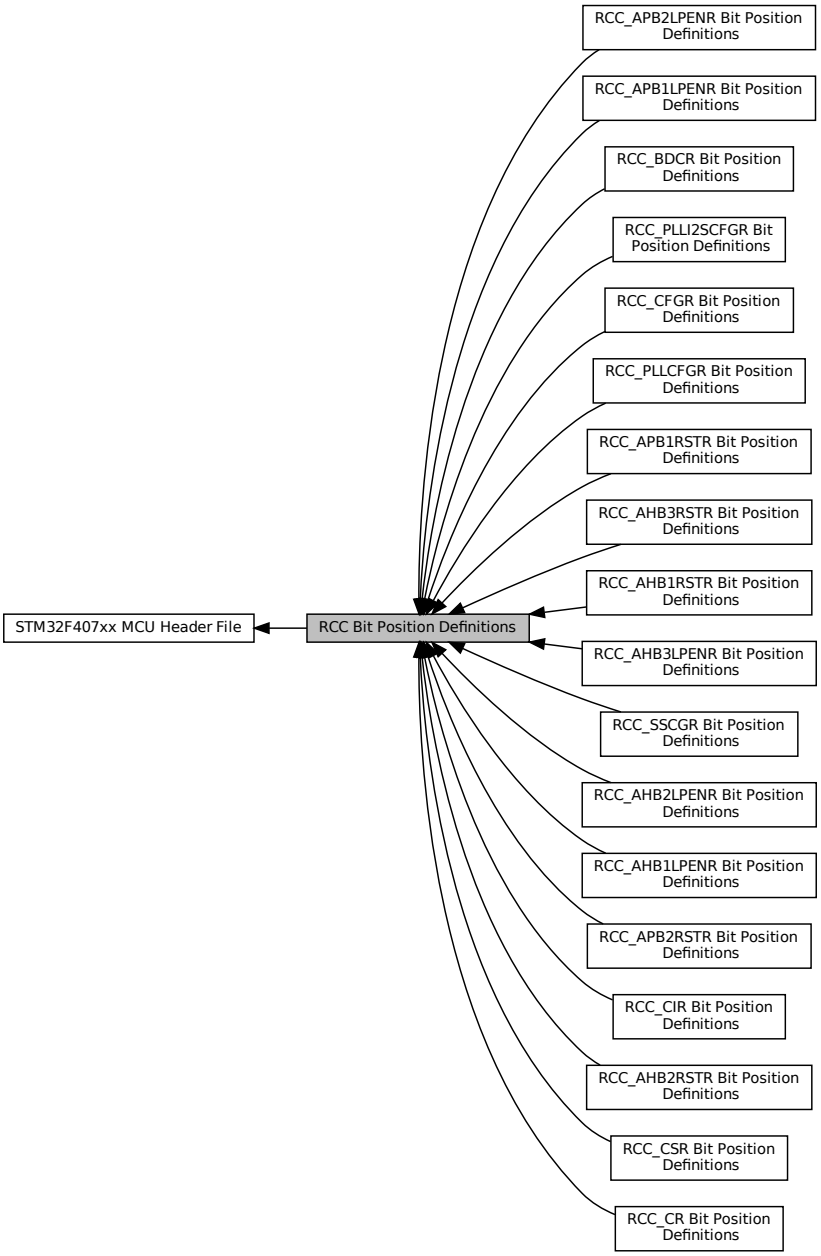
```
#define USART_GTPR_PSC 0
```

USART Prescaler Value

4.70 RCC Bit Position Definitions

Bit position definitions for various registers in the RCC peripheral.

Collaboration diagram for RCC Bit Position Definitions:



Modules

- RCC_CR Bit Position Definitions

  Bit position definitions for RCC_CR register.
- RCC_PLLCFGR Bit Position Definitions

Bit position definitions for RCC_PLLCFGR register.

- RCC_CFGR Bit Position Definitions

  Bit position definitions for RCC_CFGR register.
- RCC_CIR Bit Position Definitions

  Bit position definitions for RCC_CIR register.
- RCC_AHB1RSTR Bit Position Definitions

  Bit position definitions for RCC_AHB1RSTR register.
- RCC_AHB2RSTR Bit Position Definitions

  Bit position definitions for RCC_AHB2RSTR register.
- RCC_AHB3RSTR Bit Position Definitions

  Bit position definitions for RCC_AHB3RSTR register.
- RCC_APB1RSTR Bit Position Definitions

  Bit position definitions for RCC_APB1RSTR register.
- RCC_APB2RSTR Bit Position Definitions

  Bit position definitions for RCC_APB2RSTR register.
- RCC_AHB1LPENR Bit Position Definitions

  Bit position definitions for RCC_AHB1LPENR register.
- RCC_AHB2LPENR Bit Position Definitions

  Bit position definitions for RCC_AHB2LPENR register.
- RCC_AHB3LPENR Bit Position Definitions

  Bit position definitions for RCC_AHB3LPENR register.
- RCC_APB1LPENR Bit Position Definitions

  Bit position definitions for RCC_APB1LPENR register.
- RCC_APB2LPENR Bit Position Definitions

  Bit position definitions for RCC_APB2LPENR register.
- RCC_BDCR Bit Position Definitions

  Bit position definitions for RCC_BDCR register.
- RCC_CSR Bit Position Definitions

  Bit position definitions for RCC_CSR register.
- RCC_SSCGR Bit Position Definitions

  Bit position definitions for RCC_SSCGR register.
- RCC_PLLI2SCFGR Bit Position Definitions

  Bit position definitions for RCC_PLLI2SCFGR register.

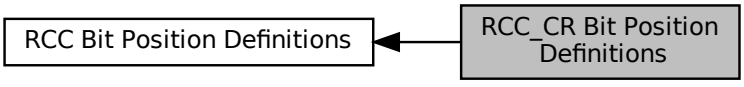
4.70.1 Detailed Description

Bit position definitions for various registers in the RCC peripheral.

4.71 RCC_CR Bit Position Definitions

Bit position definitions for RCC_CR register.

Collaboration diagram for RCC_CR Bit Position Definitions:



Macros

- #define RCC_CR_HSION 0
- #define RCC_CR_HSIRDY 1
- #define RCC_CR_HSITRIM 3
- #define RCC_CR_HSICAL 8
- #define RCC_CR_HSEON 16
- #define RCC_CR_HSERDY 17
- #define RCC_CR_HSEBYP 18
- #define RCC_CR_CSSON 19
- #define RCC_CR_PLLON 24
- #define RCC_CR_PLLRDY 25
- #define RCC_CR_PLLI2SON 26
- #define RCC_CR_PLLI2SRDY 27
- #define RCC_CR_PLLSAION 28
- #define RCC_CR_PLLSAIRDY 29

4.71.1 Detailed Description

Bit position definitions for RCC_CR register.

4.71.2 Macro Definition Documentation

4.71.2.1 RCC_CR_CSSON

```
#define RCC_CR_CSSON 19
```

Clock Security System Enable

4.71.2.2 RCC_CR_HSEBYP

```
#define RCC_CR_HSEBYP 18
```

HSE Oscillator Bypass

4.71.2.3 RCC_CR_HSEON

```
#define RCC_CR_HSEON 16
```

HSE Oscillator Enable

4.71.2.4 RCC_CR_HSERDY

```
#define RCC_CR_HSERDY 17
```

HSE Oscillator Ready

4.71.2.5 RCC_CR_HSICAL

```
#define RCC_CR_HSICAL 8
```

HSI Oscillator Calibration

4.71.2.6 RCC_CR_HSION

```
#define RCC_CR_HSION 0
```

HSI Oscillator Enable

4.71.2.7 RCC_CR_HSIRDY

```
#define RCC_CR_HSIRDY 1
```

HSI Oscillator Ready

4.71.2.8 RCC_CR_HSITRIM

```
#define RCC_CR_HSITRIM 3
```

HSI Oscillator Trimming

4.71.2.9 RCC_CR_PLLI2SON

```
#define RCC_CR_PLLI2SON 26
```

PLLI2S Enable

4.71.2.10 RCC_CR_PLLI2SRDY

```
#define RCC_CR_PLLI2SRDY 27
```

PLLI2S Ready

4.71.2.11 RCC_CR_PLLON

```
#define RCC_CR_PLLON 24
```

Main PLL Enable

4.71.2.12 RCC_CR_PLLRDY

```
#define RCC_CR_PLLRDY 25
```

Main PLL Ready

4.71.2.13 RCC_CR_PLLSAION

```
#define RCC_CR_PLLSAION 28
```

PLLSAI Enable

4.71.2.14 RCC_CR_PLLSAIRDY

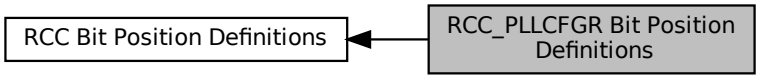
```
#define RCC_CR_PLLSAIRDY 29
```

PLLSAI Ready

4.72 RCC_PLLCFGR Bit Position Definitions

Bit position definitions for RCC_PLLCFGR register.

Collaboration diagram for RCC_PLLCFGR Bit Position Definitions:



Macros

- #define RCC_PLLCFGR_PLLM 0
- #define RCC_PLLCFGR_PLLN 6
- #define RCC_PLLCFGR_PLLP 16
- #define RCC_PLLCFGR_PLLSRC 22
- #define RCC_PLLCFGR_PLLQ 24

4.72.1 Detailed Description

Bit position definitions for RCC_PLLCFGR register.

4.72.2 Macro Definition Documentation

4.72.2.1 RCC_PLLCFGR_PLLM

```
#define RCC_PLLCFGR_PLLM 0
```

Main PLL Division Factor for PLL VCO

4.72.2.2 RCC_PLLCFGR_PLLN

```
#define RCC_PLLCFGR_PLLN 6
```

Main PLL Multiplication Factor for VCO

4.72.2.3 RCC_PLLCFGR_PLLP

```
#define RCC_PLLCFGR_PLLP 16
```

Main PLL Division Factor for Main System Clock

4.72.2.4 RCC_PLLCFGR_PLLQ

```
#define RCC_PLLCFGR_PLLQ 24
```

Main PLLQ Division Factor for PLLI2S Clock Output

4.72.2.5 RCC_PLLCFGR_PLLSRC

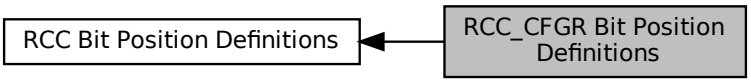
```
#define RCC_PLLCFGR_PLLSRC 22
```

Main PLL, PLLI2S, and PLLSAI Entry Clock Source

4.73 RCC_CFGR Bit Position Definitions

Bit position definitions for RCC_CFGR register.

Collaboration diagram for RCC_CFGR Bit Position Definitions:



Macros

- #define RCC_CFGR_SW 0
- #define RCC_CFGR_SWS 2
- #define RCC_CFGR_HPRE 4
- #define RCC_CFGR_PPRE1 10
- #define RCC_CFGR_PPRE2 13
- #define RCC_CFGR_RTCPRE 16
- #define RCC_CFGR_MCO1 21
- #define RCC_CFGR_I2SSRC 23
- #define RCC_CFGR_MCO1PRE 24
- #define RCC_CFGR_MCO2PRE 27
- #define RCC_CFGR_MCO2 30

4.73.1 Detailed Description

Bit position definitions for RCC_CFGR register.

4.73.2 Macro Definition Documentation

4.73.2.1 RCC_CFGR_HPRE

```
#define RCC_CFGR_HPRE 4
```

AHB Prescaler

4.73.2.2 RCC_CFGR_I2SSRC

```
#define RCC_CFGR_I2SSRC 23
```

I2S APB2 Clock Source Selection

4.73.2.3 RCC_CFGR_MCO1

```
#define RCC_CFGR_MCO1 21
```

Microcontroller Clock Output 1

4.73.2.4 RCC_CFGR_MCO1PRE

```
#define RCC_CFGR_MCO1PRE 24
```

MCO1 Prescaler

4.73.2.5 RCC_CFGR_MCO2

```
#define RCC_CFGR_MCO2 30
```

Microcontroller Clock Output 2

4.73.2.6 RCC_CFGR_MCO2PRE

```
#define RCC_CFGR_MCO2PRE 27
```

MCO2 Prescaler

4.73.2.7 RCC_CFGR_PPRE1

```
#define RCC_CFGR_PPRE1 10
```

APB1 Low-Speed Prescaler (APB1CLK)

4.73.2.8 RCC_CFGR_PPRE2

```
#define RCC_CFGR_PPRE2 13
```

APB2 High-Speed Prescaler (APB2CLK)

4.73.2.9 RCC_CFGR_RTCPRE

```
#define RCC_CFGR_RTCPRE 16
```

HSE division factor for RTC clock

4.73.2.10 RCC_CFGR_SW

```
#define RCC_CFGR_SW 0
```

System Clock Switch

4.73.2.11 RCC_CFGR_SWS

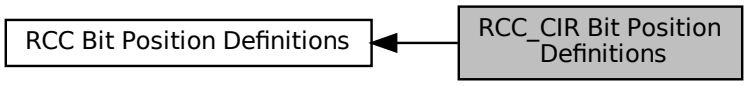
```
#define RCC_CFGR_SWS 2
```

System Clock Switch Status

4.74 RCC_CIR Bit Position Definitions

Bit position definitions for RCC_CIR register.

Collaboration diagram for RCC_CIR Bit Position Definitions:



Macros

- #define RCC_CIR_LSIRDYF 0
- #define RCC_CIR_LSERDYF 1
- #define RCC_CIR_HSIRDYF 2
- #define RCC_CIR_HSERDYF 3
- #define RCC_CIR_PLLRDYF 4
- #define RCC_CIR_PLLI2SRDYF 5
- #define RCC_CIR_PLLSAIRDYF 6
- #define RCC_CIR_CSSF 7
- #define RCC_CIR_LSIRDYIE 8
- #define RCC_CIR_LSERDYIE 9
- #define RCC_CIR_HSIRDYIE 10
- #define RCC_CIR_HSERDYIE 11
- #define RCC_CIR_PLLRDYIE 12
- #define RCC_CIR_PLLI2SRDYIE 13
- #define RCC_CIR_PLLSAIRDYIE 14
- #define RCC_CIR_LSIRDYC 16
- #define RCC_CIR_LSERDYC 17
- #define RCC_CIR_HSIRDYC 18
- #define RCC_CIR_HSERDYC 19
- #define RCC_CIR_PLLRDYC 20
- #define RCC_CIR_PLLI2SRDYC 21
- #define RCC_CIR_PLLSAIRDYC 22

4.74.1 Detailed Description

Bit position definitions for RCC_CIR register.

4.74.2 Macro Definition Documentation

4.74.2.1 RCC_CIR_CSSF

```
#define RCC_CIR_CSSF 7
```

Clock Security System Interrupt flag

4.74.2.2 RCC_CIR_HSERDYC

```
#define RCC_CIR_HSERDYC 19
```

HSE Ready Interrupt Clear

4.74.2.3 RCC_CIR_HSERDYF

```
#define RCC_CIR_HSERDYF 3
```

HSE Ready Interrupt flag

4.74.2.4 RCC_CIR_HSERDYIE

```
#define RCC_CIR_HSERDYIE 11
```

HSE Ready Interrupt Enable

4.74.2.5 RCC_CIR_HSIRDYC

```
#define RCC_CIR_HSIRDYC 18
```

HSI Ready Interrupt Clear

4.74.2.6 RCC_CIR_HSIRDYF

```
#define RCC_CIR_HSIRDYF 2
```

HSI Ready Interrupt flag

4.74.2.7 RCC_CIR_HSIRDYIE

```
#define RCC_CIR_HSIRDYIE 10
```

HSI Ready Interrupt Enable

4.74.2.8 RCC_CIR_LSERDYC

```
#define RCC_CIR_LSERDYC 17
```

LSE Ready Interrupt Clear

4.74.2.9 RCC_CIR_LSERDYF

```
#define RCC_CIR_LSERDYF 1
```

LSE Ready Interrupt flag

4.74.2.10 RCC_CIR_LSERDYIE

```
#define RCC_CIR_LSERDYIE 9
```

LSE Ready Interrupt Enable

4.74.2.11 RCC_CIR_LSIRDYC

```
#define RCC_CIR_LSIRDYC 16
```

LSI Ready Interrupt Clear

4.74.2.12 RCC_CIR_LSIRDYF

```
#define RCC_CIR_LSIRDYF 0
```

LSI Ready Interrupt flag

4.74.2.13 RCC_CIR_LSIRDYIE

```
#define RCC_CIR_LSIRDYIE 8
```

LSI Ready Interrupt Enable

4.74.2.14 RCC_CIR_PLLI2SRDYC

```
#define RCC_CIR_PLLI2SRDYC 21
```

PLLI2S Ready Interrupt Clear

4.74.2.15 RCC_CIR_PLLI2SRDYF

```
#define RCC_CIR_PLLI2SRDYF 5
```

PLLI2S Ready Interrupt flag

4.74.2.16 RCC_CIR_PLLI2SRDYIE

```
#define RCC_CIR_PLLI2SRDYIE 13
```

PLLI2S Ready Interrupt Enable

4.74.2.17 RCC_CIR_PLLRDYC

```
#define RCC_CIR_PLLRDYC 20
```

PLL Ready Interrupt Clear

4.74.2.18 RCC_CIR_PLLRDYF

```
#define RCC_CIR_PLLRDYF 4
```

PLL Ready Interrupt flag

4.74.2.19 RCC_CIR_PLLRDYIE

```
#define RCC_CIR_PLLRDYIE 12
```

PLL Ready Interrupt Enable

4.74.2.20 RCC_CIR_PLLSAIRDYC

```
#define RCC_CIR_PLLSAIRDYC 22
```

PLLSAI Ready Interrupt Clear

4.74.2.21 RCC_CIR_PLLSAIRDYF

```
#define RCC_CIR_PLLSAIRDYF 6
```

PLLSAI Ready Interrupt flag

4.74.2.22 RCC_CIR_PLLSAIRDYIE

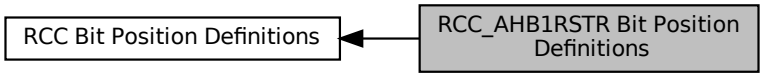
```
#define RCC_CIR_PLLSAIRDYIE 14
```

PLLSAI Ready Interrupt Enable

4.75 RCC_AHB1RSTR Bit Position Definitions

Bit position definitions for RCC_AHB1RSTR register.

Collaboration diagram for RCC_AHB1RSTR Bit Position Definitions:



Macros

- #define RCC_AHB1RSTR_GPIOA 0
- #define RCC_AHB1RSTR_GPIOB 1
- #define RCC_AHB1RSTR_GPIOC 2
- #define RCC_AHB1RSTR_GPIOD 3
- #define RCC_AHB1RSTR_GPIOE 4
- #define RCC_AHB1RSTR_GPIOF 5
- #define RCC_AHB1RSTR_GPIOG 6
- #define RCC_AHB1RSTR_GPIOH 7
- #define RCC_AHB1RSTR_GPIOI 8
- #define RCC_AHB1RSTR_CRC 12
- #define RCC_AHB1RSTR_DMA1 21
- #define RCC_AHB1RSTR_DMA2 22
- #define RCC_AHB1RSTR_ETHMAC 25
- #define RCC_AHB1RSTR_OTGHS 29
- #define RCC_AHB1RSTR_OTGHSULPI 30

4.75.1 Detailed Description

Bit position definitions for RCC_AHB1RSTR register.

4.75.2 Macro Definition Documentation

4.75.2.1 RCC_AHB1RSTR_CRC

```
#define RCC_AHB1RSTR_CRC 12
```

CRC Reset

4.75.2.2 RCC_AHB1RSTR_DMA1

```
#define RCC_AHB1RSTR_DMA1 21
```

DMA1 Reset

4.75.2.3 RCC_AHB1RSTR_DMA2

```
#define RCC_AHB1RSTR_DMA2 22
```

DMA2 Reset

4.75.2.4 RCC_AHB1RSTR_ETHMAC

```
#define RCC_AHB1RSTR_ETHMAC 25
```

Ethernet MAC Reset

4.75.2.5 RCC_AHB1RSTR_GPIOA

```
#define RCC_AHB1RSTR_GPIOA 0
```

GPIOA Reset

4.75.2.6 RCC_AHB1RSTR_GPIOB

```
#define RCC_AHB1RSTR_GPIOB 1
```

GPIOB Reset

4.75.2.7 RCC_AHB1RSTR_GPIOC

```
#define RCC_AHB1RSTR_GPIOC 2
```

GPIOC Reset

4.75.2.8 RCC_AHB1RSTR_GPIOD

```
#define RCC_AHB1RSTR_GPIOD 3
```

GPIOD Reset

4.75.2.9 RCC_AHB1RSTR_GPIOE

```
#define RCC_AHB1RSTR_GPIOE 4
```

GPIOE Reset

4.75.2.10 RCC_AHB1RSTR_GPIOF

```
#define RCC_AHB1RSTR_GPIOF 5
```

GPIOF Reset

4.75.2.11 RCC_AHB1RSTR_GPIOG

```
#define RCC_AHB1RSTR_GPIOG 6
```

GPIOG Reset

4.75.2.12 RCC_AHB1RSTR_GPIOH

```
#define RCC_AHB1RSTR_GPIOH 7
```

GPIOH Reset

4.75.2.13 RCC_AHB1RSTR_GPIOI

```
#define RCC_AHB1RSTR_GPIOI 8
```

GPIOI Reset

4.75.2.14 RCC_AHB1RSTR_OTGHS

```
#define RCC_AHB1RSTR_OTGHS 29
```

USB OTG HS Reset

4.75.2.15 RCC_AHB1RSTR_OTGHSULPI

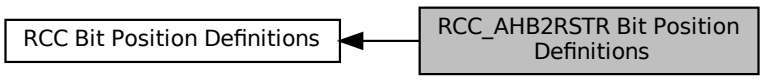
```
#define RCC_AHB1RSTR_OTGHSULPI 30
```

USB OTG HS ULPI Reset

4.76 RCC_AHB2RSTR Bit Position Definitions

Bit position definitions for RCC_AHB2RSTR register.

Collaboration diagram for RCC_AHB2RSTR Bit Position Definitions:



Macros

- #define RCC_AHB2RSTR_DCMI 0
- #define RCC_AHB2RSTR_CRYP 4
- #define RCC_AHB2RSTR_HASH 5
- #define RCC_AHB2RSTR_RNG 6
- #define RCC_AHB2RSTR_OTGFS 7

4.76.1 Detailed Description

Bit position definitions for RCC_AHB2RSTR register.

4.76.2 Macro Definition Documentation

4.76.2.1 RCC_AHB2RSTR_CRYP

```
#define RCC_AHB2RSTR_CRYP 4
```

CRYP Reset

4.76.2.2 RCC_AHB2RSTR_DCMI

```
#define RCC_AHB2RSTR_DCMI 0
```

DCMI Reset

4.76.2.3 RCC_AHB2RSTR_HASH

```
#define RCC_AHB2RSTR_HASH 5
```

HASH Reset

4.76.2.4 RCC_AHB2RSTR_OTGFS

```
#define RCC_AHB2RSTR_OTGFS 7
```

USB OTG FS Reset

4.76.2.5 RCC_AHB2RSTR_RNG

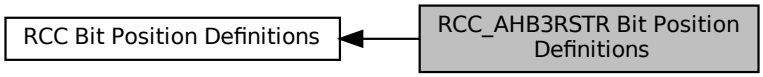
```
#define RCC_AHB2RSTR_RNG 6
```

RNG Reset

4.77 RCC_AHB3RSTR Bit Position Definitions

Bit position definitions for RCC_AHB3RSTR register.

Collaboration diagram for RCC_AHB3RSTR Bit Position Definitions:



Macros

- #define RCC_AHB3RSTR_FSMC 0

4.77.1 Detailed Description

Bit position definitions for RCC_AHB3RSTR register.

4.77.2 Macro Definition Documentation

4.77.2.1 RCC_AHB3RSTR_FSMC

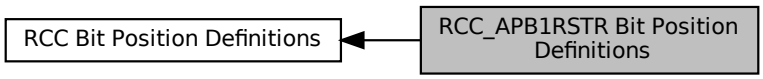
```
#define RCC_AHB3RSTR_FSMC 0
```

FSMC Reset

4.78 RCC_APB1RSTR Bit Position Definitions

Bit position definitions for RCC_APB1RSTR register.

Collaboration diagram for RCC_APB1RSTR Bit Position Definitions:



Macros

- #define RCC_APB1RSTR_TIM2 0
- #define RCC_APB1RSTR_TIM3 1
- #define RCC_APB1RSTR_TIM4 2
- #define RCC_APB1RSTR_TIM5 3
- #define RCC_APB1RSTR_TIM6 4
- #define RCC_APB1RSTR_TIM7 5
- #define RCC_APB1RSTR_TIM12 6
- #define RCC_APB1RSTR_TIM13 7
- #define RCC_APB1RSTR_TIM14 8
- #define RCC_APB1RSTR_WWDG 11
- #define RCC_APB1RSTR_SPI2 14
- #define RCC_APB1RSTR_SPI3 15
- #define RCC_APB1RSTR_USART2 17
- #define RCC_APB1RSTR_USART3 18
- #define RCC_APB1RSTR_UART4 19
- #define RCC_APB1RSTR_UART5 20
- #define RCC_APB1RSTR_I2C1 21
- #define RCC_APB1RSTR_I2C2 22
- #define RCC_APB1RSTR_I2C3 23
- #define RCC_APB1RSTR_CAN1 25
- #define RCC_APB1RSTR_CAN2 26
- #define RCC_APB1RSTR_PWR 28
- #define RCC_APB1RSTR_DAC 29
- #define RCC_APB1RSTR_UART7 30
- #define RCC_APB1RSTR_UART8 31

4.78.1 Detailed Description

Bit position definitions for RCC_APB1RSTR register.

4.78.2 Macro Definition Documentation

4.78.2.1 RCC_APB1RSTR_CAN1

```
#define RCC_APB1RSTR_CAN1 25
```

CAN1 Reset

4.78.2.2 RCC_APB1RSTR_CAN2

```
#define RCC_APB1RSTR_CAN2 26
```

CAN2 Reset

4.78.2.3 RCC_APB1RSTR_DAC

```
#define RCC_APB1RSTR_DAC 29
```

DAC Reset

4.78.2.4 RCC_APB1RSTR_I2C1

```
#define RCC_APB1RSTR_I2C1 21
```

I2C1 Reset

4.78.2.5 RCC_APB1RSTR_I2C2

```
#define RCC_APB1RSTR_I2C2 22
```

I2C2 Reset

4.78.2.6 RCC_APB1RSTR_I2C3

```
#define RCC_APB1RSTR_I2C3 23
```

I2C3 Reset

4.78.2.7 RCC_APB1RSTR_PWR

```
#define RCC_APB1RSTR_PWR 28
```

Power Interface Reset

4.78.2.8 RCC_APB1RSTR_SPI2

```
#define RCC_APB1RSTR_SPI2 14
```

SPI2 Reset

4.78.2.9 RCC_APB1RSTR_SPI3

```
#define RCC_APB1RSTR_SPI3 15
```

SPI3 Reset

4.78.2.10 RCC_APB1RSTR_TIM12

```
#define RCC_APB1RSTR_TIM12 6
```

TIM12 Reset

4.78.2.11 RCC_APB1RSTR_TIM13

```
#define RCC_APB1RSTR_TIM13 7
```

TIM13 Reset

4.78.2.12 RCC_APB1RSTR_TIM14

```
#define RCC_APB1RSTR_TIM14 8
```

TIM14 Reset

4.78.2.13 RCC_APB1RSTR_TIM2

```
#define RCC_APB1RSTR_TIM2 0
```

TIM2 Reset

4.78.2.14 RCC_APB1RSTR_TIM3

```
#define RCC_APB1RSTR_TIM3 1
```

TIM3 Reset

4.78.2.15 RCC_APB1RSTR_TIM4

```
#define RCC_APB1RSTR_TIM4 2
```

TIM4 Reset

4.78.2.16 RCC_APB1RSTR_TIM5

```
#define RCC_APB1RSTR_TIM5 3
```

TIM5 Reset

4.78.2.17 RCC_APB1RSTR_TIM6

```
#define RCC_APB1RSTR_TIM6 4
```

TIM6 Reset

4.78.2.18 RCC_APB1RSTR_TIM7

```
#define RCC_APB1RSTR_TIM7 5
```

TIM7 Reset

4.78.2.19 RCC_APB1RSTR_UART4

```
#define RCC_APB1RSTR_UART4 19
```

UART4 Reset

4.78.2.20 RCC_APB1RSTR_UART5

```
#define RCC_APB1RSTR_UART5 20
```

UART5 Reset

4.78.2.21 RCC_APB1RSTR_UART7

```
#define RCC_APB1RSTR_UART7 30
```

UART7 Reset

4.78.2.22 RCC_APB1RSTR_UART8

```
#define RCC_APB1RSTR_UART8 31
```

UART8 Reset

4.78.2.23 RCC_APB1RSTR_USART2

```
#define RCC_APB1RSTR_USART2 17
```

USART2 Reset

4.78.2.24 RCC_APB1RSTR_USART3

```
#define RCC_APB1RSTR_USART3 18
```

USART3 Reset

4.78.2.25 RCC_APB1RSTR_WWDG

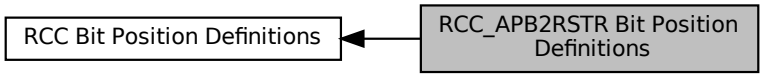
```
#define RCC_APB1RSTR_WWDG 11
```

WWDG Reset

4.79 RCC_APB2RSTR Bit Position Definitions

Bit position definitions for RCC_APB2RSTR register.

Collaboration diagram for RCC_APB2RSTR Bit Position Definitions:



Macros

- #define RCC_APB2RSTR_TIM1 0
- #define RCC_APB2RSTR_TIM8 1
- #define RCC_APB2RSTR_USART1 4
- #define RCC_APB2RSTR_USART6 5
- #define RCC_APB2RSTR_ADC 8
- #define RCC_APB2RSTR_SDIO 11
- #define RCC_APB2RSTR_SPI1 12
- #define RCC_APB2RSTR_SYSCFG 14
- #define RCC_APB2RSTR_TIM9 16
- #define RCC_APB2RSTR_TIM10 17
- #define RCC_APB2RSTR_TIM11 18

4.79.1 Detailed Description

Bit position definitions for RCC_APB2RSTR register.

4.79.2 Macro Definition Documentation

4.79.2.1 RCC_APB2RSTR_ADC

```
#define RCC_APB2RSTR_ADC 8
```

ADC Reset

4.79.2.2 RCC_APB2RSTR_SDIO

```
#define RCC_APB2RSTR_SDIO 11
```

SDIO Reset

4.79.2.3 RCC_APB2RSTR_SPI1

```
#define RCC_APB2RSTR_SPI1 12
```

SPI1 Reset

4.79.2.4 RCC_APB2RSTR_SYSCFG

```
#define RCC_APB2RSTR_SYSCFG 14
```

System Configuration Controller Reset

4.79.2.5 RCC_APB2RSTR_TIM1

```
#define RCC_APB2RSTR_TIM1 0
```

TIM1 Reset

4.79.2.6 RCC_APB2RSTR_TIM10

```
#define RCC_APB2RSTR_TIM10 17
```

TIM10 Reset

4.79.2.7 RCC_APB2RSTR_TIM11

```
#define RCC_APB2RSTR_TIM11 18
```

TIM11 Reset

4.79.2.8 RCC_APB2RSTR_TIM8

```
#define RCC_APB2RSTR_TIM8 1
```

TIM8 Reset

4.79.2.9 RCC_APB2RSTR_TIM9

```
#define RCC_APB2RSTR_TIM9 16
```

TIM9 Reset

4.79.2.10 RCC_APB2RSTR_USART1

```
#define RCC_APB2RSTR_USART1 4
```

USART1 Reset

4.79.2.11 RCC_APB2RSTR_USART6

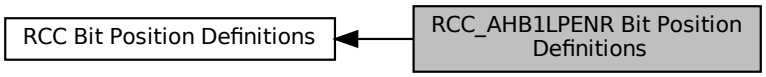
```
#define RCC_APB2RSTR_USART6 5
```

USART6 Reset

4.80 RCC_AHB1LPENR Bit Position Definitions

Bit position definitions for RCC_AHB1LPENR register.

Collaboration diagram for RCC_AHB1LPENR Bit Position Definitions:



Macros

- #define RCC_AHB1LPENR_GPIOALPEN 0
- #define RCC_AHB1LPENR_GPIOBLPEN 1
- #define RCC_AHB1LPENR_GPIOCLPEN 2
- #define RCC_AHB1LPENR_GPIODLPEN 3
- #define RCC_AHB1LPENR_GPIOELPEN 4
- #define RCC_AHB1LPENR_GPIOFLPEN 5
- #define RCC_AHB1LPENR_GPIOGLPEN 6
- #define RCC_AHB1LPENR_GPIOHLPEN 7
- #define RCC_AHB1LPENR_GPIOILPEN 8
- #define RCC_AHB1LPENR_CRCEN 12
- #define RCC_AHB1LPENR_DMA1LPEN 21
- #define RCC_AHB1LPENR_DMA2LPEN 22
- #define RCC_AHB1LPENR_ETHMACLPEN 25
- #define RCC_AHB1LPENR_ETHMACTXLPEN 26
- #define RCC_AHB1LPENR_ETHMACRXLPEN 27
- #define RCC_AHB1LPENR_ETHMACPTPLPEN 28
- #define RCC_AHB1LPENR_OTGHSLPEN 29
- #define RCC_AHB1LPENR_OTGHSHULPI 30

4.80.1 Detailed Description

Bit position definitions for RCC_AHB1LPENR register.

4.80.2 Macro Definition Documentation

4.80.2.1 RCC_AHB1LPENR_CRCEN

```
#define RCC_AHB1LPENR_CRCEN 12
```

CRC Peripheral Clock in Low Power Mode Enable

4.80.2.2 RCC_AHB1LPENR_DMA1LPEN

```
#define RCC_AHB1LPENR_DMA1LPEN 21
```

DMA1 Peripheral Clock in Low Power Mode Enable

4.80.2.3 RCC_AHB1LPENR_DMA2LPEN

```
#define RCC_AHB1LPENR_DMA2LPEN 22
```

DMA2 Peripheral Clock in Low Power Mode Enable

4.80.2.4 RCC_AHB1LPENR_ETHMACLPEN

```
#define RCC_AHB1LPENR_ETHMACLPEN 25
```

Ethernet MAC Peripheral Clock in Low Power Mode Enable

4.80.2.5 RCC_AHB1LPENR_ETHMACPTPLPEN

```
#define RCC_AHB1LPENR_ETHMACPTPLPEN 28
```

Ethernet MAC PTP Peripheral Clock in Low Power Mode Enable

4.80.2.6 RCC_AHB1LPENR_ETHMACRXLPEN

```
#define RCC_AHB1LPENR_ETHMACRXLPEN 27
```

Ethernet MAC Receive Peripheral Clock in Low Power Mode Enable

4.80.2.7 RCC_AHB1LPENR_ETHMACTXLPEN

```
#define RCC_AHB1LPENR_ETHMACTXLPEN 26
```

Ethernet MAC Transmit Peripheral Clock in Low Power Mode Enable

4.80.2.8 RCC_AHB1LPENR_GPIOALPEN

```
#define RCC_AHB1LPENR_GPIOALPEN 0
```

GPIOA Peripheral Clock in Low Power Mode Enable

4.80.2.9 RCC_AHB1LPENR_GPIOBLPEN

```
#define RCC_AHB1LPENR_GPIOBLPEN 1
```

GPIOB Peripheral Clock in Low Power Mode Enable

4.80.2.10 RCC_AHB1LPENR_GPIOCLPEN

```
#define RCC_AHB1LPENR_GPIOCLPEN 2
```

GPIOC Peripheral Clock in Low Power Mode Enable

4.80.2.11 RCC_AHB1LPENR_GPIODLPEN

```
#define RCC_AHB1LPENR_GPIODLPEN 3
```

GPIOD Peripheral Clock in Low Power Mode Enable

4.80.2.12 RCC_AHB1LPENR_GPIOELPEN

```
#define RCC_AHB1LPENR_GPIOELPEN 4
```

GPIOE Peripheral Clock in Low Power Mode Enable

4.80.2.13 RCC_AHB1LPENR_GPIOFLPEN

```
#define RCC_AHB1LPENR_GPIOFLPEN 5
```

GPIOF Peripheral Clock in Low Power Mode Enable

4.80.2.14 RCC_AHB1LPENR_GPIOGLPEN

```
#define RCC_AHB1LPENR_GPIOGLPEN 6
```

GPIOG Peripheral Clock in Low Power Mode Enable

4.80.2.15 RCC_AHB1LPENR_GPIOHLPEN

```
#define RCC_AHB1LPENR_GPIOHLPEN 7
```

GPIOH Peripheral Clock in Low Power Mode Enable

4.80.2.16 RCC_AHB1LPENR_GPIOILPEN

```
#define RCC_AHB1LPENR_GPIOILPEN 8
```

GPIOI Peripheral Clock in Low Power Mode Enable

4.80.2.17 RCC_AHB1LPENR_OTGHSHULPI

```
#define RCC_AHB1LPENR_OTGHSHULPI 30
```

USB OTG HS ULPI Peripheral Clock in Low Power Mode Enable

4.80.2.18 RCC_AHB1LPENR_OTGHSLPEN

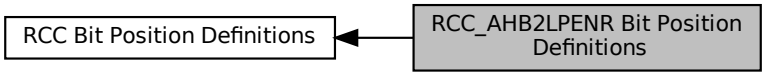
```
#define RCC_AHB1LPENR_OTGHSLPEN 29
```

USB OTG HS Peripheral Clock in Low Power Mode Enable

4.81 RCC_AHB2LPENR Bit Position Definitions

Bit position definitions for RCC_AHB2LPENR register.

Collaboration diagram for RCC_AHB2LPENR Bit Position Definitions:



Macros

- #define RCC_AHB2LPENR_DCMILPEN 0
- #define RCC_AHB2LPENR_CRYPLPEN 4
- #define RCC_AHB2LPENR_HASHLPEN 5
- #define RCC_AHB2LPENR_RNGLPEN 6
- #define RCC_AHB2LPENR_OTGFSLPEN 7

4.81.1 Detailed Description

Bit position definitions for RCC_AHB2LPENR register.

4.81.2 Macro Definition Documentation

4.81.2.1 RCC_AHB2LPENR_CRYPLPEN

```
#define RCC_AHB2LPENR_CRYPLPEN 4
```

CRYP Peripheral Clock in Low Power Mode Enable

4.81.2.2 RCC_AHB2LPENR_DCMILPEN

```
#define RCC_AHB2LPENR_DCMILPEN 0
```

DCMI Peripheral Clock in Low Power Mode Enable

4.81.2.3 RCC_AHB2LPENR_HASHLPEN

```
#define RCC_AHB2LPENR_HASHLPEN 5
```

HASH Peripheral Clock in Low Power Mode Enable

4.81.2.4 RCC_AHB2LPENR_OTGFSLPEN

```
#define RCC_AHB2LPENR_OTGFSLPEN 7
```

USB OTG FS Peripheral Clock in Low Power Mode Enable

4.81.2.5 RCC_AHB2LPENR_RNGLPEN

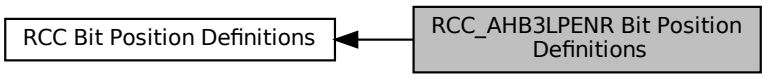
```
#define RCC_AHB2LPENR_RNGLPEN 6
```

RNG Peripheral Clock in Low Power Mode Enable

4.82 RCC_AHB3LPENR Bit Position Definitions

Bit position definitions for RCC_AHB3LPENR register.

Collaboration diagram for RCC_AHB3LPENR Bit Position Definitions:



Macros

- #define RCC_AHB3LPENR_FSMCLPEN 0

4.82.1 Detailed Description

Bit position definitions for RCC_AHB3LPENR register.

4.82.2 Macro Definition Documentation

4.82.2.1 RCC_AHB3LPENR_FSMCLPEN

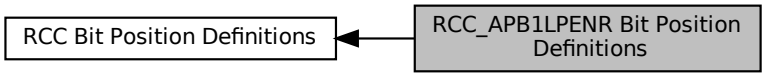
```
#define RCC_AHB3LPENR_FSMCLPEN 0
```

FSMC Peripheral Clock in Low Power Mode Enable

4.83 RCC_APB1LPENR Bit Position Definitions

Bit position definitions for RCC_APB1LPENR register.

Collaboration diagram for RCC_APB1LPENR Bit Position Definitions:



Macros

- #define RCC_APB1LPENR_TIM2LPEN 0
- #define RCC_APB1LPENR_TIM3LPEN 1
- #define RCC_APB1LPENR_TIM4LPEN 2
- #define RCC_APB1LPENR_TIM5LPEN 3
- #define RCC_APB1LPENR_TIM6LPEN 4
- #define RCC_APB1LPENR_TIM7LPEN 5
- #define RCC_APB1LPENR_TIM12LPEN 6
- #define RCC_APB1LPENR_TIM13LPEN 7
- #define RCC_APB1LPENR_TIM14LPEN 8
- #define RCC_APB1LPENR_WWDGLPEN 11
- #define RCC_APB1LPENR_SPI2LPEN 14
- #define RCC_APB1LPENR_SPI3LPEN 15
- #define RCC_APB1LPENR_USART2LPEN 17
- #define RCC_APB1LPENR_USART3LPEN 18
- #define RCC_APB1LPENR_UART4LPEN 19
- #define RCC_APB1LPENR_UART5LPEN 20
- #define RCC_APB1LPENR_I2C1LPEN 21
- #define RCC_APB1LPENR_I2C2LPEN 22
- #define RCC_APB1LPENR_I2C3LPEN 23
- #define RCC_APB1LPENR_CAN1LPEN 25
- #define RCC_APB1LPENR_CAN2LPEN 26
- #define RCC_APB1LPENR_PWRLPEN 28
- #define RCC_APB1LPENR_DACLPEN 29
- #define RCC_APB1LPENR_UART7LPEN 30
- #define RCC_APB1LPENR_UART8LPEN 31

4.83.1 Detailed Description

Bit position definitions for RCC_APB1LPENR register.

4.83.2 Macro Definition Documentation

4.83.2.1 RCC_APB1LPENR_CAN1LPEN

```
#define RCC_APB1LPENR_CAN1LPEN 25
```

CAN1 Peripheral Clock in Low Power Mode Enable

4.83.2.2 RCC_APB1LPENR_CAN2LPEN

```
#define RCC_APB1LPENR_CAN2LPEN 26
```

CAN2 Peripheral Clock in Low Power Mode Enable

4.83.2.3 RCC_APB1LPENR_DACLPEN

```
#define RCC_APB1LPENR_DACLPEN 29
```

DAC Peripheral Clock in Low Power Mode Enable

4.83.2.4 RCC_APB1LPENR_I2C1LPEN

```
#define RCC_APB1LPENR_I2C1LPEN 21
```

I2C1 Peripheral Clock in Low Power Mode Enable

4.83.2.5 RCC_APB1LPENR_I2C2LPEN

```
#define RCC_APB1LPENR_I2C2LPEN 22
```

I2C2 Peripheral Clock in Low Power Mode Enable

4.83.2.6 RCC_APB1LPENR_I2C3LPEN

```
#define RCC_APB1LPENR_I2C3LPEN 23
```

I2C3 Peripheral Clock in Low Power Mode Enable

4.83.2.7 RCC_APB1LPENR_PWRLPEN

```
#define RCC_APB1LPENR_PWRLPEN 28
```

Power Interface Peripheral Clock in Low Power Mode Enable

4.83.2.8 RCC_APB1LPENR_SPI2LPEN

```
#define RCC_APB1LPENR_SPI2LPEN 14
```

SPI2 Peripheral Clock in Low Power Mode Enable

4.83.2.9 RCC_APB1LPENR_SPI3LPEN

```
#define RCC_APB1LPENR_SPI3LPEN 15
```

SPI3 Peripheral Clock in Low Power Mode Enable

4.83.2.10 RCC_APB1LPENR_TIM12LPEN

```
#define RCC_APB1LPENR_TIM12LPEN 6
```

TIM12 Peripheral Clock in Low Power Mode Enable

4.83.2.11 RCC_APB1LPENR_TIM13LPEN

```
#define RCC_APB1LPENR_TIM13LPEN 7
```

TIM13 Peripheral Clock in Low Power Mode Enable

4.83.2.12 RCC_APB1LPENR_TIM14LPEN

```
#define RCC_APB1LPENR_TIM14LPEN 8
```

TIM14 Peripheral Clock in Low Power Mode Enable

4.83.2.13 RCC_APB1LPENR_TIM2LPEN

```
#define RCC_APB1LPENR_TIM2LPEN 0
```

TIM2 Peripheral Clock in Low Power Mode Enable

4.83.2.14 RCC_APB1LPENR_TIM3LPEN

```
#define RCC_APB1LPENR_TIM3LPEN 1
```

TIM3 Peripheral Clock in Low Power Mode Enable

4.83.2.15 RCC_APB1LPENR_TIM4LPEN

```
#define RCC_APB1LPENR_TIM4LPEN 2
```

TIM4 Peripheral Clock in Low Power Mode Enable

4.83.2.16 RCC_APB1LPENR_TIM5LPEN

```
#define RCC_APB1LPENR_TIM5LPEN 3
```

TIM5 Peripheral Clock in Low Power Mode Enable

4.83.2.17 RCC_APB1LPENR_TIM6LPEN

```
#define RCC_APB1LPENR_TIM6LPEN 4
```

TIM6 Peripheral Clock in Low Power Mode Enable

4.83.2.18 RCC_APB1LPENR_TIM7LPEN

```
#define RCC_APB1LPENR_TIM7LPEN 5
```

TIM7 Peripheral Clock in Low Power Mode Enable

4.83.2.19 RCC_APB1LPENR_UART4LPEN

```
#define RCC_APB1LPENR_UART4LPEN 19
```

UART4 Peripheral Clock in Low Power Mode Enable

4.83.2.20 RCC_APB1LPENR_UART5LPEN

```
#define RCC_APB1LPENR_UART5LPEN 20
```

UART5 Peripheral Clock in Low Power Mode Enable

4.83.2.21 RCC_APB1LPENR_UART7LPEN

```
#define RCC_APB1LPENR_UART7LPEN 30
```

UART7 Peripheral Clock in Low Power Mode Enable

4.83.2.22 RCC_APB1LPENR_UART8LPEN

```
#define RCC_APB1LPENR_UART8LPEN 31
```

UART8 Peripheral Clock in Low Power Mode Enable

4.83.2.23 RCC_APB1LPENR_USART2LPEN

```
#define RCC_APB1LPENR_USART2LPEN 17
```

USART2 Peripheral Clock in Low Power Mode Enable

4.83.2.24 RCC_APB1LPENR_USART3LPEN

```
#define RCC_APB1LPENR_USART3LPEN 18
```

USART3 Peripheral Clock in Low Power Mode Enable

4.83.2.25 RCC_APB1LPENR_WWDGLPEN

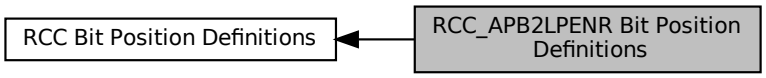
```
#define RCC_APB1LPENR_WWDGLPEN 11
```

WWDG Peripheral Clock in Low Power Mode Enable

4.84 RCC_APB2LPENR Bit Position Definitions

Bit position definitions for RCC_APB2LPENR register.

Collaboration diagram for RCC_APB2LPENR Bit Position Definitions:



Macros

- #define RCC_APB2LPENR_TIM1LPEN 0
- #define RCC_APB2LPENR_TIM8LPEN 1
- #define RCC_APB2LPENR_USART1LPEN 4
- #define RCC_APB2LPENR_USART6LPEN 5
- #define RCC_APB2LPENR_ADCLPEN 8
- #define RCC_APB2LPENR_SDIOLPEN 11
- #define RCC_APB2LPENR_SPI1LPEN 12
- #define RCC_APB2LPENR_SYSCFGLPEN 14
- #define RCC_APB2LPENR_TIM9LPEN 16
- #define RCC_APB2LPENR_TIM10LPEN 17
- #define RCC_APB2LPENR_TIM11LPEN 18

4.84.1 Detailed Description

Bit position definitions for RCC_APB2LPENR register.

4.84.2 Macro Definition Documentation

4.84.2.1 RCC_APB2LPENR_ADCLPEN

```
#define RCC_APB2LPENR_ADCLPEN 8
```

ADC Peripheral Clock in Low Power Mode Enable

4.84.2.2 RCC_APB2LPENR_SDIOLPEN

```
#define RCC_APB2LPENR_SDIOLPEN 11
```

SDIO Peripheral Clock in Low Power Mode Enable

4.84.2.3 RCC_APB2LPENR_SPI1LPEN

```
#define RCC_APB2LPENR_SPI1LPEN 12
```

SPI1 Peripheral Clock in Low Power Mode Enable

4.84.2.4 RCC_APB2LPENR_SYSCFGLPEN

```
#define RCC_APB2LPENR_SYSCFGLPEN 14
```

System Configuration Controller Peripheral Clock in Low Power Mode Enable

4.84.2.5 RCC_APB2LPENR_TIM10LPEN

```
#define RCC_APB2LPENR_TIM10LPEN 17
```

TIM10 Peripheral Clock in Low Power Mode Enable

4.84.2.6 RCC_APB2LPENR_TIM11LPEN

```
#define RCC_APB2LPENR_TIM11LPEN 18
```

TIM11 Peripheral Clock in Low Power Mode Enable

4.84.2.7 RCC_APB2LPENR_TIM1LPEN

```
#define RCC_APB2LPENR_TIM1LPEN 0
```

TIM1 Peripheral Clock in Low Power Mode Enable

4.84.2.8 RCC_APB2LPENR_TIM8LPEN

```
#define RCC_APB2LPENR_TIM8LPEN 1
```

TIM8 Peripheral Clock in Low Power Mode Enable

4.84.2.9 RCC_APB2LPENR_TIM9LPEN

```
#define RCC_APB2LPENR_TIM9LPEN 16
```

TIM9 Peripheral Clock in Low Power Mode Enable

4.84.2.10 RCC_APB2LPENR_USART1LPEN

```
#define RCC_APB2LPENR_USART1LPEN 4
```

USART1 Peripheral Clock in Low Power Mode Enable

4.84.2.11 RCC_APB2LPENR_USART6LPEN

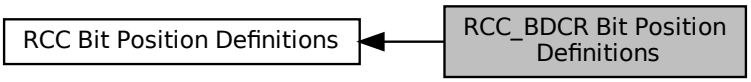
```
#define RCC_APB2LPENR_USART6LPEN 5
```

USART6 Peripheral Clock in Low Power Mode Enable

4.85 RCC_BDCR Bit Position Definitions

Bit position definitions for RCC_BDCR register.

Collaboration diagram for RCC_BDCR Bit Position Definitions:



Macros

- #define RCC_BDCR_LSEON 0
- #define RCC_BDCR_LSERDY 1
- #define RCC_BDCR_LSEBYP 2
- #define RCC_BDCR_RTCSEL 8
- #define RCC_BDCR_RTCEN 15
- #define RCC_BDCR_BDRST 16

4.85.1 Detailed Description

Bit position definitions for RCC_BDCR register.

4.85.2 Macro Definition Documentation

4.85.2.1 RCC_BDCR_BDRST

```
#define RCC_BDCR_BDRST 16
```

Backup Domain Software Reset

4.85.2.2 RCC_BDCR_LSEBYP

```
#define RCC_BDCR_LSEBYP 2
```

External Low-Speed Oscillator Bypass

4.85.2.3 RCC_BDCR_LSEON

```
#define RCC_BDCR_LSEON 0
```

External Low-Speed Oscillator Enable

4.85.2.4 RCC_BDCR_LSERDY

```
#define RCC_BDCR_LSERDY 1
```

External Low-Speed Oscillator Ready

4.85.2.5 RCC_BDCR_RTCEN

```
#define RCC_BDCR_RTCEN 15
```

RTC Clock Enable

4.85.2.6 RCC_BDCR_RTCSEL

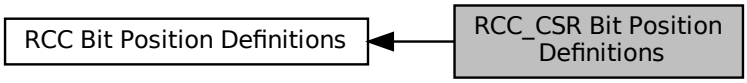
```
#define RCC_BDCR_RTCSEL 8
```

RTC Clock Source Selection

4.86 RCC_CSR Bit Position Definitions

Bit position definitions for RCC_CSR register.

Collaboration diagram for RCC_CSR Bit Position Definitions:



Macros

- #define RCC_CSR_LSION 0
- #define RCC_CSR_LSIRDY 1
- #define RCC_CSR_RMVF 24
- #define RCC_CSR_OBLRSTF 25
- #define RCC_CSR_PINRSTF 26
- #define RCC_CSR_PORRSTF 27
- #define RCC_CSR_SFTRSTF 28
- #define RCC_CSR_IWDGRSTF 29
- #define RCC_CSR_WWDGRSTF 30
- #define RCC_CSR_LPWRRSTF 31

4.86.1 Detailed Description

Bit position definitions for RCC_CSR register.

4.86.2 Macro Definition Documentation

4.86.2.1 RCC_CSR_IWDGRSTF

```
#define RCC_CSR_IWDGRSTF 29
```

Independent Watchdog Reset Flag

4.86.2.2 RCC_CSR_LPWRRSTF

```
#define RCC_CSR_LPWRRSTF 31
```

Low-Power Reset Flag

4.86.2.3 RCC_CSR_LSION

```
#define RCC_CSR_LSION 0
```

Internal Low-Speed Oscillator Enable

4.86.2.4 RCC_CSR_LSIRDY

```
#define RCC_CSR_LSIRDY 1
```

Internal Low-Speed Oscillator Ready

4.86.2.5 RCC_CSR_OBLRSTF

```
#define RCC_CSR_OBLRSTF 25
```

Option Byte Loader Reset Flag

4.86.2.6 RCC_CSR_PINRSTF

```
#define RCC_CSR_PINRSTF 26
```

PIN Reset Flag

4.86.2.7 RCC_CSR_PORRSTF

```
#define RCC_CSR_PORRSTF 27
```

POR/PDR Reset Flag

4.86.2.8 RCC_CSR_RMVF

```
#define RCC_CSR_RMVF 24
```

Remove Reset Flag

4.86.2.9 RCC_CSR_SFTRSTF

```
#define RCC_CSR_SFTRSTF 28
```

Software Reset Flag

4.86.2.10 RCC_CSR_WWDGRSTF

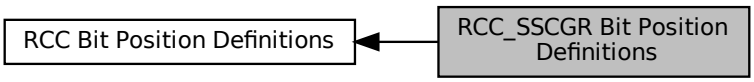
```
#define RCC_CSR_WWDGRSTF 30
```

Window Watchdog Reset Flag

4.87 RCC_SSCGR Bit Position Definitions

Bit position definitions for RCC_SSCGR register.

Collaboration diagram for RCC_SSCGR Bit Position Definitions:



Macros

- #define RCC_SSCGR_MODPER 0
- #define RCC_SSCGR_INCSTEP 13
- #define RCC_SSCGR_SPREADSEL 15
- #define RCC_SSCGR_SSCGEN 31

4.87.1 Detailed Description

Bit position definitions for RCC_SSCGR register.

4.87.2 Macro Definition Documentation

4.87.2.1 RCC_SSCGR_INCSTEP

```
#define RCC_SSCGR_INCSTEP 13
```

Increase Step

4.87.2.2 RCC_SSCGR_MODPER

```
#define RCC_SSCGR_MODPER 0
```

Modulation Period

4.87.2.3 RCC_SSCGR_SPREADSEL

```
#define RCC_SSCGR_SPREADSEL 15
```

Spread Select

4.87.2.4 RCC_SSCGR_SSCGEN

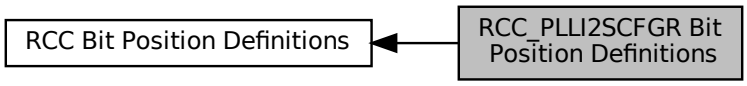
```
#define RCC_SSCGR_SSCGEN 31
```

Spread Spectrum Clock Generation Enable

4.88 RCC_PLLI2SCFGR Bit Position Definitions

Bit position definitions for RCC_PLLI2SCFGR register.

Collaboration diagram for RCC_PLLI2SCFGR Bit Position Definitions:



Macros

- #define RCC_PLLI2SCFGR_PLLI2SN 6
- #define RCC_PLLI2SCFGR_PLLI2SR 28

4.88.1 Detailed Description

Bit position definitions for RCC_PLLI2SCFGR register.

4.88.2 Macro Definition Documentation

4.88.2.1 RCC_PLLI2SCFGR_PLLI2SN

```
#define RCC_PLLI2SCFGR_PLLI2SN 6
```

PLLI2S N Factor

4.88.2.2 RCC_PLLI2SCFGR_PLLI2SR

```
#define RCC_PLLI2SCFGR_PLLI2SR 28
```

PLLI2S R Factor

4.89 Generic Macros

Generic macros for enabling/disabling, setting/resetting, and handling flags.

Collaboration diagram for Generic Macros:

Macros

- #define ENABLE 1
- #define DISABLE 0
- #define SET ENABLE
- #define RESET DISABLE
- #define GPIO_PIN_SET SET
- #define GPIO_PIN_RESET RESET
- #define FLAG_SET SET
- #define FLAG_RESET RESET

4.89.1 Detailed Description

Generic macros for enabling/disabling, setting/resetting, and handling flags.

4.89.2 Macro Definition Documentation

4.89.2.1 DISABLE

```
#define DISABLE 0
```

Disable macro

4.89.2.2 ENABLE

```
#define ENABLE 1
```

Enable macro

4.89.2.3 FLAG_RESET

```
#define FLAG_RESET RESET
```

Flag reset macro

4.89.2.4 FLAG_SET

```
#define FLAG_SET SET
```

Flag set macro

4.89.2.5 GPIO_PIN_RESET

```
#define GPIO_PIN_RESET RESET
```

GPIO Pin reset macro

4.89.2.6 GPIO_PIN_SET

```
#define GPIO_PIN_SET SET
```

GPIO Pin set macro

4.89.2.7 RESET

```
#define RESET DISABLE
```

Reset macro

4.89.2.8 SET

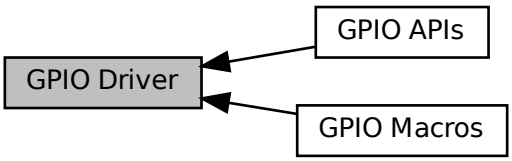
```
#define SET ENABLE
```

Set macro

4.90 GPIO Driver

GPIO driver APIs for STM32F407xx MCU.

Collaboration diagram for GPIO Driver:



Modules

- GPIO Macros

  Macros for GPIO configuration and settings.
- GPIO APIs

  APIs supported by the GPIO driver.

Classes

- struct GPIO_PinConfig_t

  Configuration structure for GPIO pin.
- struct GPIO_Handle_t

  Handle structure for GPIO pin.

Variables

- uint32_t GPIO_PinConfig_t::GPIO_PinNumber
- uint32_t GPIO_PinConfig_t::GPIO_PinMode
- uint32_t GPIO_PinConfig_t::GPIO_PinSpeed
- uint32_t GPIO_PinConfig_t::GPIO_PinPuPdControl
- uint32_t GPIO_PinConfig_t::GPIO_PinPinOPType
- uint32_t GPIO_PinConfig_t::GPIO_PinAltFunMode
- GPIO_RegDef_t * GPIO_Handle_t::pGPIOx
- GPIO_PinConfig_t GPIO_Handle_t::GPIO_PinConfig

4.90.1 Detailed Description

GPIO driver APIs for STM32F407xx MCU.

4.90.2 Variable Documentation

4.90.2.1 GPIO_PinAltFunMode

`uint32_t GPIO_PinConfig_t::GPIO_PinAltFunMode`

Specifies the alternate function mode of the GPIO pin.

4.90.2.2 GPIO_PinConfig

`GPIO_PinConfig_t GPIO_Handle_t::GPIO_PinConfig`

Holds GPIO pin configuration settings.

4.90.2.3 GPIO_PinMode

`uint32_t GPIO_PinConfig_t::GPIO_PinMode`

Specifies the mode of the GPIO pin.

4.90.2.4 GPIO_PinNumber

`uint32_t GPIO_PinConfig_t::GPIO_PinNumber`

Specifies the GPIO pin number.

4.90.2.5 GPIO_PinPinOPType

`uint32_t GPIO_PinConfig_t::GPIO_PinPinOPType`

Specifies the output type of the GPIO pin.

4.90.2.6 GPIO_PinPuPdControl

`uint32_t GPIO_PinConfig_t::GPIO_PinPuPdControl`

Specifies the pull-up/pull-down configuration for the GPIO pin.

4.90.2.7 GPIO_PinSpeed

`uint32_t GPIO_PinConfig_t::GPIO_PinSpeed`

Specifies the speed of the GPIO pin.

4.90.2.8 pGPIOx

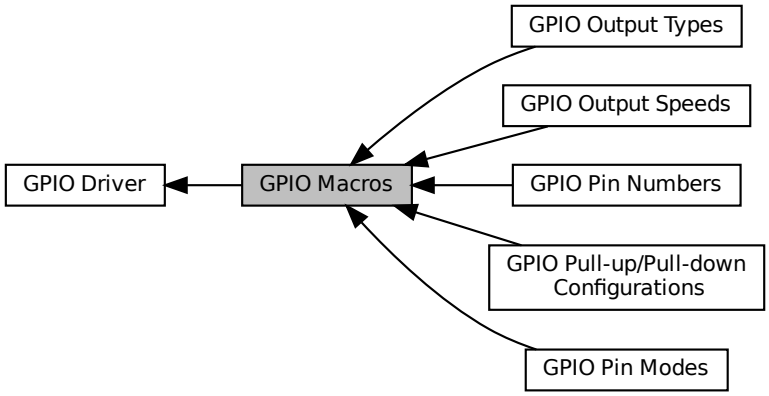
```
GPIO_RegDef_t* GPIO_Handle_t::pGPIOx
```

Holds the base address of the GPIO port to which the pin belongs.

4.91 GPIO Macros

Macros for GPIO configuration and settings.

Collaboration diagram for GPIO Macros:



Modules

- GPIO Pin Numbers

  Possible GPIO pin numbers.
- GPIO Pin Modes

  Possible GPIO pin modes.
- GPIO Output Speeds

  Possible GPIO pin output speeds.
- GPIO Pull-up/Pull-down Configurations

  Possible GPIO pin pull-up and pull-down configurations.
- GPIO Output Types

  Possible GPIO pin output types.

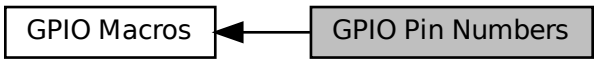
4.91.1 Detailed Description

Macros for GPIO configuration and settings.

4.92 GPIO Pin Numbers

Possible GPIO pin numbers.

Collaboration diagram for GPIO Pin Numbers:



Macros

- #define **GPIO_PIN_NO_0** 0
- #define **GPIO_PIN_NO_1** 1
- #define **GPIO_PIN_NO_2** 2
- #define **GPIO_PIN_NO_3** 3
- #define **GPIO_PIN_NO_4** 4
- #define **GPIO_PIN_NO_5** 5
- #define **GPIO_PIN_NO_6** 6
- #define **GPIO_PIN_NO_7** 7
- #define **GPIO_PIN_NO_8** 8
- #define **GPIO_PIN_NO_9** 9
- #define **GPIO_PIN_NO_10** 10
- #define **GPIO_PIN_NO_11** 11
- #define **GPIO_PIN_NO_12** 12
- #define **GPIO_PIN_NO_13** 13
- #define **GPIO_PIN_NO_14** 14
- #define **GPIO_PIN_NO_15** 15

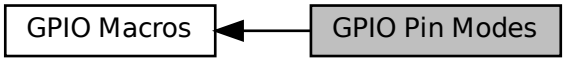
4.92.1 Detailed Description

Possible GPIO pin numbers.

4.93 GPIO Pin Modes

Possible GPIO pin modes.

Collaboration diagram for GPIO Pin Modes:



Macros

- #define GPIO_MODE_IN 0
- #define GPIO_MODE_OUT 1
- #define GPIO_MODE_ALTFN 2
- #define GPIO_MODE_ANALOG 3
- #define GPIO_MODE_IT_FT 4
- #define GPIO_MODE_IT_RT 5
- #define GPIO_MODE_IT_RFT 6

4.93.1 Detailed Description

Possible GPIO pin modes.

4.93.2 Macro Definition Documentation

4.93.2.1 GPIO_MODE_ALTFN

```
#define GPIO_MODE_ALTFN 2
```

GPIO Alternate Function mode.

4.93.2.2 GPIO_MODE_ANALOG

```
#define GPIO_MODE_ANALOG 3
```

GPIO Analog mode.

4.93.2.3 GPIO_MODE_IN

```
#define GPIO_MODE_IN 0
```

GPIO Input mode.

4.93.2.4 GPIO_MODE_IT_FT

```
#define GPIO_MODE_IT_FT 4
```

GPIO Interrupt Falling-Edge Trigger mode.

4.93.2.5 GPIO_MODE_IT_RFT

```
#define GPIO_MODE_IT_RFT 6
```

GPIO Interrupt Rising-Falling Edge Trigger mode.

4.93.2.6 GPIO_MODE_IT_RT

```
#define GPIO_MODE_IT_RT 5
```

GPIO Interrupt Rising-Edge Trigger mode.

4.93.2.7 GPIO_MODE_OUT

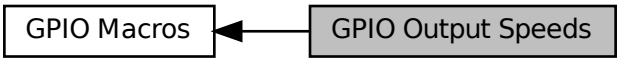
```
#define GPIO_MODE_OUT 1
```

GPIO Output mode.

4.94 GPIO Output Speeds

Possible GPIO pin output speeds.

Collaboration diagram for GPIO Output Speeds:



Macros

- #define GPIO_SPEED_LOW 0
- #define GPIO_SPEED_MEDIUM 1
- #define GPIO_SPEED_FAST 2
- #define GPIO_SPEED_HIGH 3

4.94.1 Detailed Description

Possible GPIO pin output speeds.

4.94.2 Macro Definition Documentation

4.94.2.1 GPIO_SPEED_FAST

```
#define GPIO_SPEED_FAST 2
```

GPIO Output speed Fast.

4.94.2.2 GPIO_SPEED_HIGH

```
#define GPIO_SPEED_HIGH 3
```

GPIO Output speed High.

4.94.2.3 GPIO_SPEED_LOW

```
#define GPIO_SPEED_LOW 0
```

GPIO Output speed Low.

4.94.2.4 GPIO_SPEED_MEDIUM

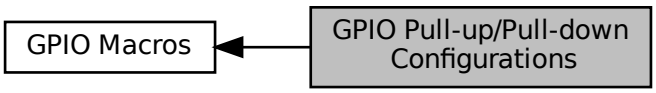
```
#define GPIO_SPEED_MEDIUM 1
```

GPIO Output speed Medium.

4.95 GPIO Pull-up/Pull-down Configurations

Possible GPIO pin pull-up and pull-down configurations.

Collaboration diagram for GPIO Pull-up/Pull-down Configurations:



Macros

- #define GPIO_NO_PUPD 0
- #define GPIO_PIN_PU 1
- #define GPIO_PIN_PD 2

4.95.1 Detailed Description

Possible GPIO pin pull-up and pull-down configurations.

4.95.2 Macro Definition Documentation

4.95.2.1 GPIO_NO_PUPD

```
#define GPIO_NO_PUPD 0
```

No pull-up/pull-down configuration.

4.95.2.2 GPIO_PIN_PD

```
#define GPIO_PIN_PD 2
```

GPIO Pull-down configuration.

4.95.2.3 GPIO_PIN_PU

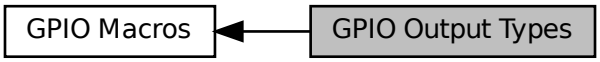
```
#define GPIO_PIN_PU 1
```

GPIO Pull-up configuration.

4.96 GPIO Output Types

Possible GPIO pin output types.

Collaboration diagram for GPIO Output Types:



Macros

- #define GPIO_OP_TYPE_PP 0
- #define GPIO_OP_TYPE_OD 1

4.96.1 Detailed Description

Possible GPIO pin output types.

4.96.2 Macro Definition Documentation

4.96.2.1 GPIO_OP_TYPE_OD

```
#define GPIO_OP_TYPE_OD 1
```

GPIO Output type Open-Drain mode.

4.96.2.2 GPIO_OP_TYPE_PP

```
#define GPIO_OP_TYPE_PP 0
```

GPIO Output type Push-Pull mode.

4.97 GPIO APIs

APIs supported by the GPIO driver.

Collaboration diagram for GPIO APIs:

Functions

- void GPIO_PeripheralClockControl (GPIO_RegDef_t *pGPIOx, uint8_t EnorDi)
  Enables or disables the peripheral clock for the GPIO port.
- void GPIO_Init (GPIO_Handle_t *pGPIOHandle)
  Initializes the GPIO port pin according to the configuration.
- void GPIO_DeInit (GPIO_RegDef_t *pGPIOx)
  Deinitializes the GPIO port.
- uint8_t GPIO_ReadFromInputPin (GPIO_RegDef_t *pGPIOx, uint8_t PinNumber)
  Reads a value from a specific GPIO pin.
- uint16_t GPIO_ReadFromInputPort (GPIO_RegDef_t *pGPIOx)
  Reads a value from the entire GPIO port.
- void GPIO_WriteToOutputPin (GPIO_RegDef_t *pGPIOx, uint16_t PinNumber, uint8_t value)
  Writes a value to a specific GPIO pin.
- void GPIO_WriteToOutputPort (GPIO_RegDef_t *pGPIOx, uint16_t value)
  Writes a value to the entire GPIO port.
- void GPIO_ToggleOutputPin (GPIO_RegDef_t *pGPIOx, uint16_t PinNumber)
  Toggles the output value of a specific GPIO pin.
- void GPIO_IRQInterruptConfig (uint8_t IRQNumber, uint8_t EnorDi)
  Configures the IRQ for a specific GPIO pin.
- void GPIO_IRQPriorityConfig (uint8_t IRQNumber, uint32_t IRQpriority)
  Configures the priority for a specific IRQ.
- void GPIO_IRQHandling (uint16_t PinNumber)
  Handles the IRQ for a specific GPIO pin.

4.97.1 Detailed Description

APIs supported by the GPIO driver.

4.97.2 Function Documentation

4.97.2.1 GPIO_DeInit()

```
void GPIO_DeInit (
            GPIO_RegDef_t * pGPIOx )
```

Deinitializes the GPIO port.

Parameters

pGPIOx	Pointer to the GPIO peripheral.

Deinitializes the GPIO port.

Parameters

pGPIOx	Pointer to GPIO port

4.97.2.2 GPIO_Init()

```
void GPIO_Init (
            GPIO_Handle_t * pGPIOHandle )
```

Initializes the GPIO port pin according to the configuration.

Parameters

pGPIOHandle	Pointer to the GPIO handle structure.

Initializes the GPIO port pin according to the configuration.

Parameters

pGPIOHandle	Pointer to GPIO_Handle_t structure

4.97.2.3 GPIO_IRQHandling()

```
void GPIO_IRQHandling (
            uint16_t PinNumber )
```

Handles the IRQ for a specific GPIO pin.

Parameters

PinNumber	GPIO pin number.

Handles the IRQ for a specific GPIO pin.

Parameters

PinNumber	GPIO pin number

4.97.2.4 GPIO_IRQInterruptConfig()

```
void GPIO_IRQInterruptConfig (
            uint8_t IRQNumber,
            uint8_t EnorDi )
```

Configures the IRQ for a specific GPIO pin.

Parameters

IRQNumber	IRQ number.
EnorDi	ENABLE to enable IRQ, DISABLE to disable IRQ.

Configures the IRQ for a specific GPIO pin.

Parameters

IRQNumber	IRQ number
EnorDi	ENABLE or DISABLE macros

4.97.2.5 GPIO_IRQPriorityConfig()

```
void GPIO_IRQPriorityConfig (
            uint8_t IRQNumber,
            uint32_t IRQPriority )
```

Configures the priority for a specific IRQ.

Parameters

IRQNumber	IRQ number.
IRQpriority	Priority value.

Configures the priority for a specific IRQ.

Parameters

IRQNumber	IRQ number
IRQpriority	Priority of the IRQ

4.97.2.6 GPIO_PeripheralClockControl()

```
void GPIO_PeripheralClockControl (
             GPIO_RegDef_t * pGPIOx,
             uint8_t EnorDi )
```

Enables or disables the peripheral clock for the GPIO port.

Parameters

pGPIOx	Pointer to the GPIO peripheral.
EnorDi	ENABLE to enable clock, DISABLE to disable clock.

Enables or disables the peripheral clock for the GPIO port.

Parameters

pGPIOx	Pointer to GPIO port
EnorDi	ENABLE or DISABLE macros

4.97.2.7 GPIO_ReadFromInputPin()

```
uint8_t GPIO_ReadFromInputPin (
             GPIO_RegDef_t * pGPIOx,
             uint8_t PinNumber )
```

Reads a value from a specific GPIO pin.

Parameters

pGPIOx	Pointer to the GPIO peripheral.
PinNumber	GPIO pin number.

Returns

uint8_t 1 if the pin is set, 0 if the pin is reset.

Reads a value from a specific GPIO pin.

Parameters

pGPIOx	Pointer to GPIO port
PinNumber	GPIO pin number

Returns

uint8_t Value of the pin (0 or 1)

4.97.2.8 GPIO_ReadFromInputPort()

```
uint16_t GPIO_ReadFromInputPort (
            GPIO_RegDef_t * pGPIOx )
```

Reads a value from the entire GPIO port.

Parameters

pGPIOx	Pointer to the GPIO peripheral.

Returns

uint16_t Content of the input data register.

Reads a value from the entire GPIO port.

Parameters

pGPIOx	Pointer to GPIO port

Returns

uint16_t Value of the GPIO port

4.97.2.9 GPIO_ToggleOutputPin()

```
void GPIO_ToggleOutputPin (
            GPIO_RegDef_t * pGPIOx,
            uint16_t PinNumber )
```

Toggles the output value of a specific GPIO pin.

Parameters

pGPIOx	Pointer to the GPIO peripheral.
PinNumber	GPIO pin number.

Toggles the output value of a specific GPIO pin.

Parameters

pGPIOx	Pointer to GPIO port
PinNumber	GPIO pin number

4.97.2.10 GPIO_WriteToOutputPin()

```
void GPIO_WriteToOutputPin (
            GPIO_RegDef_t * pGPIOx,
            uint16_t PinNumber,
            uint8_t value )
```

Writes a value to a specific GPIO pin.

Parameters

pGPIOx	Pointer to the GPIO peripheral.
PinNumber	GPIO pin number.
value	Value to write (1 or 0).

Writes a value to a specific GPIO pin.

Parameters

pGPIOx	Pointer to GPIO port
PinNumber	GPIO pin number
value	Value to be written (SET or RESET)

4.97.2.11 GPIO_WriteToOutputPort()

```
void GPIO_WriteToOutputPort (
            GPIO_RegDef_t * pGPIOx,
            uint16_t value )
```

Writes a value to the entire GPIO port.

Parameters

pGPIOx	Pointer to the GPIO peripheral.
value	Value to write.

Writes a value to the entire GPIO port.

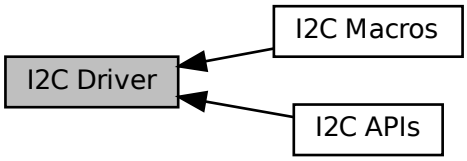
Parameters

pGPIOx	Pointer to GPIO port
value	Value to be written to the GPIO port

4.98 I2C Driver

I2C driver APIs for STM32F407xx MCU.

Collaboration diagram for I2C Driver:



Modules

- I2C Macros

  Macros related to I2C configuration, flags, and events.
- I2C APIs

  APIs supported by the I2C driver.

Classes

- struct I2C_Config_t

  Configuration structure for I2C peripheral.
- struct I2C_Handle_t

  Handle structure for I2C peripheral.

Variables

- uint32_t **I2C_Config_t::I2C_SCLspeed**
- uint8_t **I2C_Config_t::I2C_DeviceAdress**
- uint8_t **I2C_Config_t::I2C_ACKControl**
- uint8_t **I2C_Config_t::I2C_FMDutyCycle**
- I2C_RegDef_t * **I2C_Handle_t::pI2Cx**
- I2C_Config_t **I2C_Handle_t::I2C_Config**
- uint8_t * **I2C_Handle_t::pTxBuffer**
- uint8_t * **I2C_Handle_t::pRxBuffer**
- uint32_t **I2C_Handle_t::TxLen**
- uint32_t **I2C_Handle_t::RxLen**
- uint8_t **I2C_Handle_t::TxRxState**
- uint8_t **I2C_Handle_t::DevAddr**
- uint32_t **I2C_Handle_t::RxSize**
- uint8_t **I2C_Handle_t::Sr**

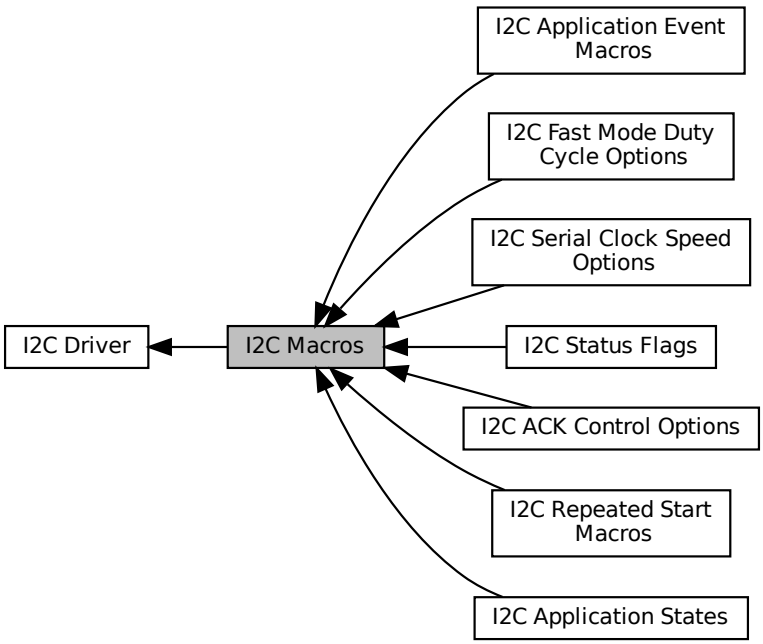
4.98.1 Detailed Description

I2C driver APIs for STM32F407xx MCU.

4.99 I2C Macros

Macros related to I2C configuration, flags, and events.

Collaboration diagram for I2C Macros:



Modules

- I2C Application States

  Possible states of the I2C application.
- I2C Serial Clock Speed Options

  Options for I2C serial clock speed.
- I2C ACK Control Options

  Options for controlling the ACK mechanism in I2C communication.
- I2C Fast Mode Duty Cycle Options

  Options for fast mode duty cycle in I2C communication.
- I2C Status Flags

  Flags indicating various status conditions in I2C communication.
- I2C Repeated Start Macros

  Macros for enabling or disabling repeated start conditions in I2C communication.
- I2C Application Event Macros

  Macros for I2C application events.

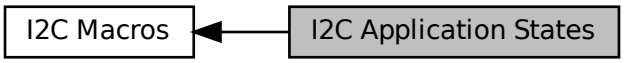
4.99.1 Detailed Description

Macros related to I2C configuration, flags, and events.

4.100 I2C Application States

Possible states of the I2C application.

Collaboration diagram for I2C Application States:



Macros

- #define I2C_READY 0
- #define I2C_BUSY_IN_RX 1
- #define I2C_BUSY_IN_TX 2

4.100.1 Detailed Description

Possible states of the I2C application.

4.100.2 Macro Definition Documentation

4.100.2.1 I2C_BUSY_IN_RX

```
#define I2C_BUSY_IN_RX 1
```

I2C application is busy receiving data.

4.100.2.2 I2C_BUSY_IN_TX

```
#define I2C_BUSY_IN_TX 2
```

I2C application is busy transmitting data.

4.100.2.3 I2C_READY

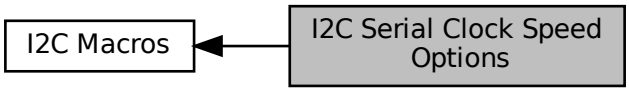
```
#define I2C_READY 0
```

I2C application is ready for communication.

4.101 I2C Serial Clock Speed Options

Options for I2C serial clock speed.

Collaboration diagram for I2C Serial Clock Speed Options:



Macros

- #define I2C_SC_SPEED_SM 100000
- #define I2C_SC_SPEED_FM4K 400000
- #define I2C_SC_SPEED_FM2K 200000

4.101.1 Detailed Description

Options for I2C serial clock speed.

4.101.2 Macro Definition Documentation

4.101.2.1 I2C_SC_SPEED_FM2K

```
#define I2C_SC_SPEED_FM2K 200000
```

Fast mode serial clock speed, up to 200 KHz.

4.101.2.2 I2C_SC_SPEED_FM4K

```
#define I2C_SC_SPEED_FM4K 400000
```

Fast mode serial clock speed, up to 400 KHz.

4.101.2.3 I2C_SC_SPEED_SM

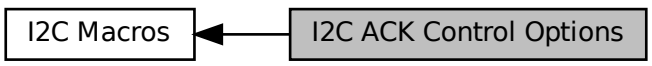
```
#define I2C_SC_SPEED_SM 100000
```

Standard mode serial clock speed, up to 100 KHz.

4.102 I2C ACK Control Options

Options for controlling the ACK mechanism in I2C communication.

Collaboration diagram for I2C ACK Control Options:



Macros

- #define I2C_ACK_ENABLE 1
- #define I2C_ACK_DISABLE 0

4.102.1 Detailed Description

Options for controlling the ACK mechanism in I2C communication.

4.102.2 Macro Definition Documentation

4.102.2.1 I2C_ACK_DISABLE

```
#define I2C_ACK_DISABLE 0
```

Disable ACK mechanism (default behavior).

4.102.2.2 I2C_ACK_ENABLE

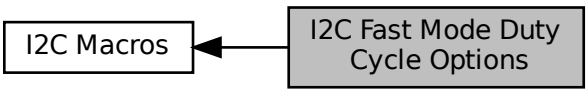
```
#define I2C_ACK_ENABLE 1
```

Enable ACK mechanism.

4.103 I2C Fast Mode Duty Cycle Options

Options for fast mode duty cycle in I2C communication.

Collaboration diagram for I2C Fast Mode Duty Cycle Options:



Macros

- #define I2C_FM_DUTY_2 0
- #define I2C_FM_DUTY_16_9 1

4.103.1 Detailed Description

Options for fast mode duty cycle in I2C communication.

4.103.2 Macro Definition Documentation

4.103.2.1 I2C_FM_DUTY_16_9

```
#define I2C_FM_DUTY_16_9 1
```

Fast mode duty cycle option: Tlow/THigh = 16/9.

4.103.2.2 I2C_FM_DUTY_2

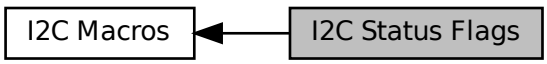
```
#define I2C_FM_DUTY_2 0
```

Fast mode duty cycle option: Tlow/THigh = 2.

4.104 I2C Status Flags

Flags indicating various status conditions in I2C communication.

Collaboration diagram for I2C Status Flags:



Macros

- #define I2C_FLAG_SB (1 << I2C_SR1_SB)
- #define I2C_FLAG_ADDR (1 << I2C_SR1_ADDR)
- #define I2C_FLAG_BTF (1 << I2C_SR1_BTF)
- #define I2C_FLAG_ADD10 (1 << I2C_SR1_ADD10)
- #define I2C_FLAG_STOPF (1 << I2C_SR1_STOPF)
- #define I2C_FLAG_RxNE (1 << I2C_SR1_RxNE)
- #define I2C_FLAG_TxE (1 << I2C_SR1_TxE)
- #define I2C_FLAG_BERR (1 << I2C_SR1_BERR)
- #define I2C_FLAG_ARLO (1 << I2C_SR1_ARLO)
- #define I2C_FLAG_AF (1 << I2C_SR1_AF)
- #define I2C_FLAG_OVR (1 << I2C_SR1_OVR)
- #define I2C_FLAG_PECERR (1 << I2C_SR1_PECERR)
- #define I2C_FLAG_TIMEOUT (1 << I2C_SR1_TIMEOUT)
- #define I2C_FLAG_SMBALERT (1 << I2C_SR1_SMBALERT)

4.104.1 Detailed Description

Flags indicating various status conditions in I2C communication.

4.104.2 Macro Definition Documentation

4.104.2.1 I2C_FLAG_ADD10

```
#define I2C_FLAG_ADD10 (1 << I2C_SR1_ADD10)
```

10-bit header sent flag.

4.104.2.2 I2C_FLAG_ADDR

```
#define I2C_FLAG_ADDR (1 << I2C_SR1_ADDR)
```

Address sent flag.

4.104.2.3 I2C_FLAG_AF

```
#define I2C_FLAG_AF (1 << I2C_SR1_AF)
```

Acknowledge failure flag.

4.104.2.4 I2C_FLAG_ARLO

```
#define I2C_FLAG_ARLO (1 << I2C_SR1_ARLO)
```

Arbitration lost error flag.

4.104.2.5 I2C_FLAG_BERR

```
#define I2C_FLAG_BERR (1 << I2C_SR1_BERR)
```

Bus error flag.

4.104.2.6 I2C_FLAG_BTF

```
#define I2C_FLAG_BTF (1 << I2C_SR1_BTF)
```

Byte transfer finished flag.

4.104.2.7 I2C_FLAG_OVR

```
#define I2C_FLAG_OVR (1 << I2C_SR1_OVR)
```

Overrun/underrun error flag.

4.104.2.8 I2C_FLAG_PECERR

```
#define I2C_FLAG_PECERR (1 << I2C_SR1_PECERR)
```

PEC error in reception flag.

4.104.2.9 I2C_FLAG_RxNE

```
#define I2C_FLAG_RxNE (1 << I2C_SR1_RxNE)
```

Receive data register not empty flag.

4.104.2.10 I2C_FLAG_SB

```
#define I2C_FLAG_SB (1 << I2C_SR1_SB)
```

Start bit flag.

4.104.2.11 I2C_FLAG_SMBALERT

```
#define I2C_FLAG_SMBALERT (1 << I2C_SR1_SMBALERT)
```

SMBus alert flag.

4.104.2.12 I2C_FLAG_STOPF

```
#define I2C_FLAG_STOPF (1 << I2C_SR1_STOPF)
```

Stop detection flag.

4.104.2.13 I2C_FLAG_TIMEOUT

```
#define I2C_FLAG_TIMEOUT (1 << I2C_SR1_TIMEOUT)
```

Timeout error flag.

4.104.2.14 I2C_FLAG_TxE

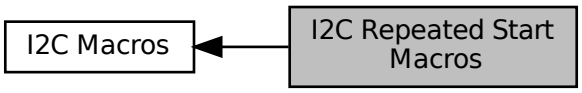
```
#define I2C_FLAG_TxE (1 << I2C_SR1_TxE)
```

Transmit data register empty flag.

4.105 I2C Repeated Start Macros

Macros for enabling or disabling repeated start conditions in I2C communication.

Collaboration diagram for I2C Repeated Start Macros:



Macros

- #define I2C_ENABLE_SR SET
- #define I2C_DISABLE_SR RESET

4.105.1 Detailed Description

Macros for enabling or disabling repeated start conditions in I2C communication.

4.105.2 Macro Definition Documentation

4.105.2.1 I2C_DISABLE_SR

```
#define I2C_DISABLE_SR RESET
```

Macro to disable repeated start condition (RESET).

4.105.2.2 I2C_ENABLE_SR

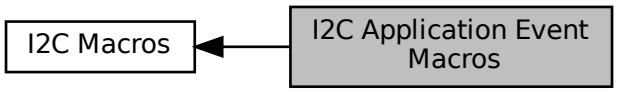
```
#define I2C_ENABLE_SR SET
```

Macro to enable repeated start condition (SET).

4.106 I2C Application Event Macros

Macros for I2C application events.

Collaboration diagram for I2C Application Event Macros:



Macros

- #define I2C_EV_TX_CMPLT 0
- #define I2C_EV_RX_CMPLT 1
- #define I2C_EV_STOP 2
- #define I2C_ERROR_BERR 3
- #define I2C_ERROR_ARLO 4
- #define I2C_ERROR_AF 5
- #define I2C_ERROR_OVR 6
- #define I2C_ERROR_TIMEOUT 7
- #define I2C_EV_DATA_REQ 8
- #define I2C_EV_DATA_RCV 9

4.106.1 Detailed Description

Macros for I2C application events.

4.106.2 Macro Definition Documentation

4.106.2.1 I2C_ERROR_AF

```
#define I2C_ERROR_AF 5
```

Event: Acknowledge failure.

4.106.2.2 I2C_ERROR_ARLO

```
#define I2C_ERROR_ARLO 4
```

Event: Arbitration lost error.

4.106.2.3 I2C_ERROR_BERR

```
#define I2C_ERROR_BERR 3
```

Event: Bus error.

4.106.2.4 I2C_ERROR_OVR

```
#define I2C_ERROR_OVR 6
```

Event: Overrun/underrun error.

4.106.2.5 I2C_ERROR_TIMEOUT

```
#define I2C_ERROR_TIMEOUT 7
```

Event: Timeout error.

4.106.2.6 I2C_EV_DATA_RCV

```
#define I2C_EV_DATA_RCV 9
```

Event: Data reception.

4.106.2.7 I2C_EV_DATA_REQ

```
#define I2C_EV_DATA_REQ 8
```

Event: Data request.

4.106.2.8 I2C_EV_RX_CMPLT

```
#define I2C_EV_RX_CMPLT 1
```

Event: Reception complete.

4.106.2.9 I2C_EV_STOP

```
#define I2C_EV_STOP 2
```

Event: Stop condition.

4.106.2.10 I2C_EV_TX_CMPLT

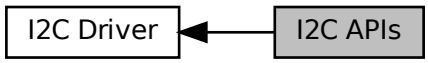
```
#define I2C_EV_TX_CMPLT 0
```

Event: Transmission complete.

4.107 I2C APIs

APIs supported by the I2C driver.

Collaboration diagram for I2C APIs:



Functions

- void I2C_PeriClockControl (I2C_RegDef_t *pI2Cx, uint8_t EnorDi)

  Controls the peripheral clock of the I2C peripheral.
- void I2C_Init (I2C_Handle_t *pI2CHandle)

  Initializes the I2C peripheral.
- void I2C_DeInit (I2C_RegDef_t *pI2Cx)

  Deinitializes the I2C peripheral.
- void I2C_MasterSendData (I2C_Handle_t *pI2CHandle, uint8_t *pTxBuffer, uint32_t Len, uint8_t SlaveAddr, uint8_t SR)

  Sends data in master mode over the I2C bus.
- void I2C_MasterReceiveData (I2C_Handle_t *pI2CHandle, uint8_t *pRxBuffer, uint8_t Len, uint8_t Slave↩ Addr, uint8_t SR)

  Receives data in master mode over the I2C bus.
- void I2C_SlaveSendData (I2C_RegDef_t *pI2Cx, uint8_t data)

  Sends a single byte of data in slave mode over the I2C bus.
- uint8_t I2C_SlaveReceiveData (I2C_RegDef_t *pI2Cx)

  Receives a single byte of data in slave mode over the I2C bus.
- uint8_t I2C_MasterSendDataIT (I2C_Handle_t *pI2CHandle, uint8_t *pTxBuffer, uint32_t Len, uint8_↩ t SlaveAddr, uint8_t SR)

  Sends data in master mode over the I2C bus with interrupt support.
- uint8_t I2C_MasterReceiveDataIT (I2C_Handle_t *pI2CHandle, uint8_t *pRxBuffer, uint8_t Len, uint8_↩ t SlaveAddr, uint8_t SR)

  Receives data in master mode over the I2C bus with interrupt support.
- void I2C_CloseSendData (I2C_Handle_t *pI2CHandle)

  Closes the transmission process.
- void I2C_CloseReceiveData (I2C_Handle_t *pI2CHandle)

  Closes the reception process.
- void I2C_IRQInterruptConfig (uint8_t IRQNumber, uint8_t EnorDi)

  Configures IRQ interrupt for the I2C peripheral.
- void I2C_IRQPriorityConfig (uint8_t IRQNumber, uint32_t IRQPriority)

  Configures IRQ priority for the I2C peripheral.
- void I2C_EV_IRQHandling (I2C_Handle_t *pI2CHandle)

  Handles I2C event interrupt.
- void I2C_ER_IRQHandling (I2C_Handle_t *pI2CHandle)

Handles I2C error interrupt.

- void I2C_PeripheralControl (I2C_RegDef_t *pI2Cx, uint8_t EnorDi)

  Controls peripheral operation in master mode.
- uint8_t I2C_GetFlagStatus (I2C_RegDef_t *pI2Cx, uint32_t FlagName)

  Retrieves the flag status of the specified I2C flag.
- void I2C_ManageAcking (I2C_RegDef_t *pI2Cx, uint8_t EnorDi)

  Manages ACKing during I2C communication.
- void I2C_GenerateStopCondition (I2C_RegDef_t *pI2Cx)

  Generates a stop condition on the I2C bus.
- void I2C_SlaveEnableDisableCallbackEvents (I2C_RegDef_t *pI2Cx, uint8_t EnorDi)

  Enables or disables callback events for the I2C slave.
- void I2C_ApplicationEventCallback (I2C_Handle_t *pI2CHandle, uint8_t AppEv)

  Handles application events for the I2C communication.

4.107.1 Detailed Description

APIs supported by the I2C driver.

4.107.2 Function Documentation

4.107.2.1 I2C_ApplicationEventCallback()

```
void I2C_ApplicationEventCallback (
             I2C_Handle_t * pI2CHandle,
             uint8_t AppEv )
```

Handles application events for the I2C communication.

Parameters

pI2CHandle	Pointer to the I2C handle structure.
AppEv	Application event to be handled.

Handles application events for the I2C communication.

This function serves as a callback that can be overridden by the application to handle I2C-related events. It is called when specific events occur during I2C communication and allows the application to take custom actions.

Parameters

pI2CHandle	Pointer to the I2C handle structure.
AppEv	I2C application event.

Note

This is a weak implementation, and the application can override this function to provide custom event handling.

4.107.2.2 I2C_CloseReceiveData()

```
void I2C_CloseReceiveData (
            I2C_Handle_t * pI2CHandle )
```

Closes the reception process.

Parameters

pI2CHandle	Pointer to the I2C handle structure.

Closes the reception process.

Parameters

pI2CHandle	Pointer to the I2C handle structure.

4.107.2.3 I2C_CloseSendData()

```
void I2C_CloseSendData (
            I2C_Handle_t * pI2CHandle )
```

Closes the transmission process.

Parameters

pI2CHandle	Pointer to the I2C handle structure.

Closes the transmission process.

Parameters

pI2CHandle	Pointer to the I2C handle structure.

4.107.2.4 I2C_DeInit()

```
void I2C_DeInit (
            I2C_RegDef_t * pI2Cx )
```

Deinitializes the I2C peripheral.

Parameters

pI2Cx	Pointer to the I2C peripheral.
pI2Cx	Pointer to the I2C peripheral register structure.

4.107.2.5 I2C_ER_IRQHandling()

```
void I2C_ER_IRQHandling (
            I2C_Handle_t * pI2CHandle )
```

Handles I2C error interrupt.

Parameters

pI2CHandle	Pointer to the I2C handle structure.

Handles I2C error interrupt.

Parameters

pI2CHandle	Pointer to the I2C handle structure.

4.107.2.6 I2C_EV_IRQHandling()

```
void I2C_EV_IRQHandling (
            I2C_Handle_t * pI2CHandle )
```

Handles I2C event interrupt.

Parameters

pI2CHandle	Pointer to the I2C handle structure.

Handles I2C event interrupt.

Parameters

pI2CHandle	Pointer to the I2C handle structure.

4.107.2.7 I2C_GenerateStopCondition()

```
void I2C_GenerateStopCondition (
             I2C_RegDef_t * pI2Cx )
```

Generates a stop condition on the I2C bus.

Parameters

pI2Cx	Pointer to the I2C peripheral.

Generates a stop condition on the I2C bus.

Parameters

pI2Cx	Pointer to the I2C peripheral register structure.

4.107.2.8 I2C_GetFlagStatus()

```
uint8_t I2C_GetFlagStatus (
             I2C_RegDef_t * pI2Cx,
             uint32_t FlagName )
```

Retrieves the flag status of the specified I2C flag.

Parameters

pI2Cx	Pointer to the I2C peripheral.
FlagName	Flag to check.

Returns

uint8_t FLAG_SET if flag is set, FLAG_RESET if not.

Retrieves the flag status of the specified I2C flag.

Parameters

pI2Cx	Pointer to the I2C peripheral.
FlagName	The flag to check status for.

Returns

uint8_t FLAG_SET if the flag is set, FLAG_RESET if not set.

4.107.2.9 I2C_Init()

```
void I2C_Init (
            I2C_Handle_t * pI2CHandle )
```

Initializes the I2C peripheral.

Parameters

pI2CHandle	Pointer to the I2C handle structure.

Initializes the I2C peripheral.

Parameters

pI2CHandle	Pointer to the I2C handle structure.

4.107.2.10 I2C_IRQInterruptConfig()

```
void I2C_IRQInterruptConfig (
            uint8_t IRQNumber,
            uint8_t EnorDi )
```

Configures IRQ interrupt for the I2C peripheral.

Parameters

IRQNumber	IRQ number.
EnorDi	ENABLE to enable the IRQ, DISABLE to disable.

Configures IRQ interrupt for the I2C peripheral.

Parameters

IRQNumber	IRQ number to be configured.
EnorDi	Enable or disable the IRQ (ENABLE or DISABLE).

4.107.2.11 I2C_IRQPriorityConfig()

```
void I2C_IRQPriorityConfig (
            uint8_t IRQNumber,
            uint32_t IRQPriority )
```

Configures IRQ priority for the I2C peripheral.

Parameters

IRQNumber	IRQ number.
IRQPriority	IRQ priority value.

Configures IRQ priority for the I2C peripheral.

Parameters

IRQNumber	IRQ number to be configured.
IRQPriority	Priority to be set for the IRQ.

4.107.2.12 I2C_ManageAcking()

```
void I2C_ManageAcking (
            I2C_RegDef_t * pI2Cx,
            uint8_t EnorDi )
```

Manages ACKing during I2C communication.

Parameters

pI2Cx	Pointer to the I2C peripheral.
EnorDi	I2C_ACK_ENABLE to enable ACKing, I2C_ACK_DISABLE to disable.

Manages ACKing during I2C communication.

Parameters

pI2Cx	Pointer to the I2C peripheral.
EnorDi	I2C_ACK_ENABLE to enable acknowledgment, I2C_ACK_DISABLE to disable acknowledgment.

4.107.2.13 I2C_MasterReceiveData()

```
void I2C_MasterReceiveData (
            I2C_Handle_t * pI2CHandle,
            uint8_t * pRxBuffer,
            uint8_t Len,
            uint8_t SlaveAddr,
            uint8_t SR )
```

Receives data in master mode over the I2C bus.

Parameters

pI2CHandle	Pointer to the I2C handle structure.
pRxBuffer	Pointer to the receive buffer.
Len	Length of data to be received.
SlaveAddr	Slave address to communicate with.
SR	Repeated start setting.

Receives data in master mode over the I2C bus.

Parameters

pI2CHandle	Pointer to the I2C handle structure.
pRxBuffer	Pointer to the receive buffer.
Len	Number of bytes to receive.
SlaveAddr	Address of the slave device.
SR	Generate Stop Condition after the transaction (ENABLE or DISABLE).

4.107.2.14 I2C_MasterReceiveDataIT()

```
uint8_t I2C_MasterReceiveDataIT (
             I2C_Handle_t * pI2CHandle,
             uint8_t * pRxBuffer,
             uint8_t Len,
             uint8_t SlaveAddr,
             uint8_t Sr )
```

Receives data in master mode over the I2C bus with interrupt support.

Parameters

pI2CHandle	Pointer to the I2C handle structure.
pRxBuffer	Pointer to the receive buffer.
Len	Length of data to be received.
SlaveAddr	Slave address to communicate with.
SR	Repeated start setting.

Returns

uint8_t Status flag.

Receives data in master mode over the I2C bus with interrupt support.

Parameters

pI2CHandle	Pointer to the I2C handle structure.
pRxBuffer	Pointer to the receive buffer.

Parameters

Len	Number of bytes to receive.
SlaveAddr	Address of the slave device.
Sr	Generate Repeated Start condition (ENABLE or DISABLE).

Returns

uint8_t Current state of the I2C peripheral (I2C_BUSY_IN_TX or I2C_BUSY_IN_RX).

4.107.2.15 I2C_MasterSendData()

```
void I2C_MasterSendData (
             I2C_Handle_t * pI2CHandle,
             uint8_t * pTxBuffer,
             uint32_t Len,
             uint8_t SlaveAddr,
             uint8_t SR )
```

Sends data in master mode over the I2C bus.

Parameters

pI2CHandle	Pointer to the I2C handle structure.
pTxBuffer	Pointer to the transmit buffer.
Len	Length of data to be sent.
SlaveAddr	Slave address to communicate with.
SR	Repeated start setting.

Sends data in master mode over the I2C bus.

Parameters

pI2CHandle	Pointer to the I2C handle structure.
pTxBuffer	Pointer to the transmit buffer.
Len	Number of bytes to transmit.
SlaveAddr	Address of the slave device.
SR	Generate Stop Condition after the transaction: (ENABLE or DISABLE) "the repeated start condition".

4.107.2.16 I2C_MasterSendDataIT()

```
uint8_t I2C_MasterSendDataIT (
             I2C_Handle_t * pI2CHandle,
```

```
            uint8_t * pTxBuffer,
            uint32_t Len,
            uint8_t SlaveAddr,
            uint8_t Sr )
```

Sends data in master mode over the I2C bus with interrupt support.

Parameters

pI2CHandle	Pointer to the I2C handle structure.
pTxBuffer	Pointer to the transmit buffer.
Len	Length of data to be sent.
SlaveAddr	Slave address to communicate with.
SR	Repeated start setting.

Returns

uint8_t Status flag.

Sends data in master mode over the I2C bus with interrupt support.

Parameters

pI2CHandle	Pointer to the I2C handle structure.
pTxBuffer	Pointer to the transmit buffer.
Len	Number of bytes to transmit.
SlaveAddr	Address of the slave device.
Sr	Generate Repeated Start condition (ENABLE or DISABLE).

Returns

uint8_t Current state of the I2C peripheral (I2C_BUSY_IN_TX or I2C_BUSY_IN_RX).

4.107.2.17 I2C_PeriClockControl()

```
void I2C_PeriClockControl (
            I2C_RegDef_t * pI2Cx,
            uint8_t EnorDi )
```

Controls the peripheral clock of the I2C peripheral.

Parameters

pI2Cx	Pointer to the I2C peripheral.
Enor↩ Di	ENABLE to enable the clock, DISABLE to disable.

Controls the peripheral clock of the I2C peripheral.

Parameters

pI2Cx	Pointer to the I2C peripheral register structure.
EnorDi	Enable or disable action. Use ENABLE to enable and DISABLE to disable.

4.107.2.18 I2C_PeripheralControl()

```
void I2C_PeripheralControl (
             I2C_RegDef_t * pI2Cx,
             uint8_t EnorDi )
```

Controls peripheral operation in master mode.

Parameters

pI2Cx	Pointer to the I2C peripheral.
EnorDi	ENABLE to enable, DISABLE to disable.

Controls peripheral operation in master mode.

Parameters

pI2Cx	Pointer to the I2C peripheral.
EnorDi	ENABLE to enable, DISABLE to disable.

4.107.2.19 I2C_SlaveEnableDisableCallbackEvents()

```
void I2C_SlaveEnableDisableCallbackEvents (
             I2C_RegDef_t * pI2Cx,
             uint8_t EnorDi )
```

Enables or disables callback events for the I2C slave.

Parameters

pI2Cx	Pointer to the I2C peripheral.
EnorDi	ENABLE to enable callback events, DISABLE to disable.

4.107.2.20 I2C_SlaveReceiveData()

```
uint8_t I2C_SlaveReceiveData (
            I2C_RegDef_t * pI2Cx )
```

Receives a single byte of data in slave mode over the I2C bus.

Parameters

pI2Cx	Pointer to the I2C peripheral.

Returns

uint8_t Received data.

Receives a single byte of data in slave mode over the I2C bus.

Parameters

pI2Cx	Pointer to the I2C peripheral.

Returns

uint8_t Received data.

4.107.2.21 I2C_SlaveSendData()

```
void I2C_SlaveSendData (
            I2C_RegDef_t * pI2Cx,
            uint8_t data )
```

Sends a single byte of data in slave mode over the I2C bus.

Parameters

pI2Cx	Pointer to the I2C peripheral.
data	Data to be sent.

Sends a single byte of data in slave mode over the I2C bus.

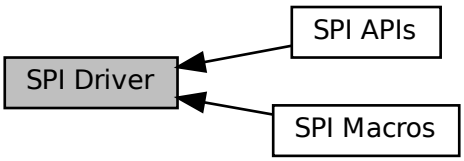
Parameters

pI2Cx	Pointer to the I2C peripheral.
data	Data to be sent.

4.108 SPI Driver

SPI driver APIs for STM32F407xx MCU.

Collaboration diagram for SPI Driver:



Modules

- SPI APIs

  APIs supported by the SPI driver.
- SPI Macros

  Macros related to SPI configuration, flags, and events.

Classes

- struct SPI_Config_t

  Configuration structure for SPI peripheral.
- struct SPI_Handle_t

  Handle structure for SPI peripheral.

Variables

- uint8_t SPI_Config_t::SPI_DeviceMode
- uint8_t SPI_Config_t::SPI_BusConfig
- uint8_t SPI_Config_t::SPI_SclkSpeed
- uint8_t SPI_Config_t::SPI_DFF
- uint8_t SPI_Config_t::SPI_CPOL
- uint8_t SPI_Config_t::SPI_CPHA
- uint8_t SPI_Config_t::SPI_SSM
- SPI_RegDef_t * SPI_Handle_t::pSPIx
- SPI_Config_t SPI_Handle_t::SPI_Config
- uint8_t * SPI_Handle_t::pTxBuffer
- uint8_t * SPI_Handle_t::pRxBuffer
- uint32_t SPI_Handle_t::TxLen
- uint32_t SPI_Handle_t::RxLen
- uint8_t SPI_Handle_t::TxState
- uint8_t SPI_Handle_t::RxState

4.108.1 Detailed Description

SPI driver APIs for STM32F407xx MCU.

4.108.2 Variable Documentation

4.108.2.1 pRxBuffer

```
uint8_t* SPI_Handle_t::pRxBuffer
```

Pointer to the receive buffer.

4.108.2.2 pSPIx

```
SPI_RegDef_t* SPI_Handle_t::pSPIx
```

Pointer to the SPI peripheral's base address.

4.108.2.3 pTxBuffer

```
uint8_t* SPI_Handle_t::pTxBuffer
```

Pointer to the transmit buffer.

4.108.2.4 RxLen

```
uint32_t SPI_Handle_t::RxLen
```

Length of data to be received.

4.108.2.5 RxState

```
uint8_t SPI_Handle_t::RxState
```

Receive state (used for interrupt-driven reception).

4.108.2.6 SPI_BusConfig

```
uint8_t SPI_Config_t::SPI_BusConfig
```

SPI bus configuration (Full-duplex/Half-duplex/...).

4.108.2.7 SPI_Config

`SPI_Config_t SPI_Handle_t::SPI_Config`

SPI configuration structure.

4.108.2.8 SPI_CPHA

`uint8_t SPI_Config_t::SPI_CPHA`

SPI clock phase (Data capture on rising/falling edge).

4.108.2.9 SPI_CPOL

`uint8_t SPI_Config_t::SPI_CPOL`

SPI clock polarity (Idle state polarity).

4.108.2.10 SPI_DeviceMode

`uint8_t SPI_Config_t::SPI_DeviceMode`

SPI device mode (Master/Slave).

4.108.2.11 SPI_DFF

`uint8_t SPI_Config_t::SPI_DFF`

SPI data frame format (8/16-bit data frame).

4.108.2.12 SPI_SclkSpeed

`uint8_t SPI_Config_t::SPI_SclkSpeed`

SPI serial clock speed (Prescaler selection).

4.108.2.13 SPI_SSM

`uint8_t SPI_Config_t::SPI_SSM`

SPI software slave management (Enable/Disable).

4.108.2.14 TxLen

`uint32_t SPI_Handle_t::TxLen`

Length of data to be transmitted.

4.108.2.15 TxState

```
uint8_t SPI_Handle_t::TxState
```

Transmit state (used for interrupt-driven transmission).

4.109 SPI APIs

APIs supported by the SPI driver.

Collaboration diagram for SPI APIs:

Functions

- void SPI_PeripheralClockControl (SPI_RegDef_t *pSPIx, uint8_t EnorDi)

  Enables or disables the peripheral clock for the SPI port.
- void SPI_PeripheralControl (SPI_RegDef_t *pSPIx, uint8_t EnorDi)

  Enables or disables the SPI peripheral.
- void SPI_Init (SPI_Handle_t *pSPIHandle)

  Initializes the SPI port pin according to the configuration.
- void SPI_DeInit (SPI_RegDef_t *pSPIx)

  Deinitializes the SPI port.
- uint8_t SPI_GetFlagStatus (SPI_RegDef_t *pSPIx, uint8_t FlagName)

  Get the status of a specific SPI flag.
- void SPI_SendData (SPI_RegDef_t *pSPIx, uint8_t *pTxBuffer, uint32_t Len)

  Sends data over SPI.
- void SPI_ReceiveData (SPI_RegDef_t *pSPIx, uint8_t *pRxBuffer, uint32_t Len)

  Receives data over SPI.
- uint8_t SPI_SendDataIT (SPI_Handle_t *pSPIHandle, uint8_t *pTxBuffer, uint32_t Len)

  Sends data over SPI using interrupt-driven communication.
- uint8_t SPI_ReceiveDataIT (SPI_Handle_t *pSPIHandle, uint8_t *pRxBuffer, uint32_t Len)

  Receives data over SPI using interrupt-driven communication.
- void SPI_IRQinterruptConfig (uint8_t IRQNumber, uint8_t EnorDi)

  Configures the IRQ for a specific SPI pin.
- void SPI_IRQperiorityConfig (uint8_t IRQNumber, uint32_t IRQpriority)

  Configures the priority for a specific IRQ.
- void SPI_IRQHandling (SPI_Handle_t *pSPIHandle)

  Handles the IRQ for a specific SPI pin.
- void SPI_SSIConfig (SPI_RegDef_t *pSPIx, uint8_t EnorDi)

  Enables or disables the Software Slave Management (SSI) configuration for the SPI peripheral.

- void SPI_SSOEConfig (SPI_RegDef_t *pSPIx, uint8_t EnorDi)

  Enables or disables the SPI Slave Select Output Enable (SSOE) configuration for the SPI peripheral.
- void SPI_ClearOVRFlag (SPI_RegDef_t *pSPIx)

  Clears the overrun error (OVR) flag in the SPI peripheral.
- void SPI_CloseTransmission (SPI_Handle_t *pSPIHandle)

  Closes the transmission operation on the SPI peripheral.
- void SPI_CloseReception (SPI_Handle_t *pSPIHandle)

  Closes the reception operation on the SPI peripheral.
- void SPI_ApplicationEventCallback (SPI_Handle_t *pSPIHandle, uint8_t AppEv)

  SPI Application Event Callback.

4.109.1 Detailed Description

APIs supported by the SPI driver.

4.109.2 Function Documentation

4.109.2.1 SPI_ApplicationEventCallback()

```
void SPI_ApplicationEventCallback (
             SPI_Handle_t * pSPIHandle,
             uint8_t AppEv )
```

SPI Application Event Callback.

This function serves as a callback that can be overridden by the application to handle SPI-related events. It is called when specific events occur during SPI communication and allows the application to take custom actions.

Parameters

pSPIHandle	Pointer to the SPI handle structure.
AppEv	The SPI application event that occurred (e.g., transmission complete, reception complete).

Note

This is a weak implementation, and the application can override this function to provide custom event handling.

4.109.2.2 SPI_ClearOVRFlag()

```
void SPI_ClearOVRFlag (
             SPI_RegDef_t * pSPIx )
```

Clears the overrun error (OVR) flag in the SPI peripheral.

This function clears the overrun error flag in the status register of the SPI peripheral. Overrun errors occur when new data is received before the previous data is read, causing data loss.

Parameters

pSPIx	Pointer to the SPI peripheral for which the overrun error flag should be cleared.

4.109.2.3 SPI_CloseReception()

```
void SPI_CloseReception (
            SPI_Handle_t * pSPIHandle )
```

Closes the reception operation on the SPI peripheral.

This function is used to close the reception operation on the SPI peripheral after all the data has been received. It disables the receive buffer not empty interrupt and sets the reception state to idle.

Parameters

pSPIHandle	Pointer to the SPI handle structure.

4.109.2.4 SPI_CloseTransmission()

```
void SPI_CloseTransmission (
            SPI_Handle_t * pSPIHandle )
```

Closes the transmission operation on the SPI peripheral.

This function is used to close the transmission operation on the SPI peripheral after all the data has been transmitted. It disables the transmit buffer empty interrupt and sets the transmission state to idle.

Parameters

pSPIHandle	Pointer to the SPI handle structure.

4.109.2.5 SPI_DeInit()

```
void SPI_DeInit (
            SPI_RegDef_t * pSPIx )
```

Deinitializes the SPI port.

Parameters

pSPIx	Pointer to the SPI peripheral.

4.109.2.6 SPI_GetFlagStatus()

```
uint8_t SPI_GetFlagStatus (
            SPI_RegDef_t * pSPIx,
            uint8_t FlagName )
```

Get the status of a specific SPI flag.

Parameters

pSPIx	Pointer to the SPI peripheral.
FlagName	Flag to check (e.g., SPI_TXE_FLAG, SPI_RXNE_FLAG).

Returns

FLAG_SET if the flag is set, FLAG_RESET if not.

4.109.2.7 SPI_Init()

```
void SPI_Init (
            SPI_Handle_t * pSPIHandle )
```

Initializes the SPI port pin according to the configuration.

Parameters

pSPIHandle	Pointer to the SPI handle structure.

4.109.2.8 SPI_IRQHandling()

```
void SPI_IRQHandling (
            SPI_Handle_t * pSPIHandle )
```

Handles the IRQ for a specific SPI pin.

Parameters

PinNumber	SPI pin number.

4.109.2.9 SPI_IRQinterruptConfig()

```
void SPI_IRQinterruptConfig (
            uint8_t IRQNumber,
            uint8_t EnorDi )
```

Configures the IRQ for a specific SPI pin.

Parameters

IRQNumber	IRQ number.
EnorDi	ENABLE to enable IRQ, DISABLE to disable IRQ.

4.109.2.10 SPI_IRQperiorityConfig()

```
void SPI_IRQperiorityConfig (
            uint8_t IRQNumber,
            uint32_t IRQpriority )
```

Configures the priority for a specific IRQ.

Parameters

IRQNumber	IRQ number.
IRQpriority	Priority value.

4.109.2.11 SPI_PeripheralClockControl()

```
void SPI_PeripheralClockControl (
            SPI_RegDef_t * pSPIx,
            uint8_t EnorDi )
```

Enables or disables the peripheral clock for the SPI port.

Parameters

pSPIx	Pointer to the SPI peripheral.
Enor↩ Di	ENABLE to enable clock, DISABLE to disable clock.

4.109.2.12 SPI_PeripheralControl()

```
void SPI_PeripheralControl (
```

```
            SPI_RegDef_t * pSPIx,
            uint8_t EnorDi )
```

Enables or disables the SPI peripheral.

This function allows you to enable or disable the SPI peripheral. When enabled, the SPI port can send and receive data. When disabled, the SPI port is inactive and cannot send or receive data.

Parameters

pSPIx	Pointer to the SPI peripheral's register structure.
EnorDi	ENABLE to enable the SPI peripheral, DISABLE to disable it.

Note

Disabling the SPI peripheral will halt ongoing SPI transactions.

4.109.2.13 SPI_ReceiveData()

```
void SPI_ReceiveData (
            SPI_RegDef_t * pSPIx,
            uint8_t * pRxBuffer,
            uint32_t Len )
```

Receives data over SPI.

Parameters

pSPIx	Pointer to the SPI peripheral.
pRxBuffer	Pointer to the receive buffer.
Len	Length of data to be received.

4.109.2.14 SPI_ReceiveDataIT()

```
uint8_t SPI_ReceiveDataIT (
            SPI_Handle_t * pSPIHandle,
            uint8_t * pRxBuffer,
            uint32_t Len )
```

Receives data over SPI using interrupt-driven communication.

Parameters

pSPIHandle	Pointer to the SPI handle structure.
pRxBuffer	Pointer to the receive buffer.
Len	Length of data to be received.

Note

This function sets up and initiates the reception of data over SPI using interrupts. The data reception will be handled asynchronously, and the user should implement the necessary interrupt handler to process received data.

Returns

- SPI_READY: If the SPI is ready for communication and the data reception is successfully initiated.
- SPI_BUSY_IN_RX: If a previous reception is still ongoing.
- SPI_ERROR_INVALID_ARG: If the input parameters are invalid.

4.109.2.15 SPI_SendData()

```
void SPI_SendData (
            SPI_RegDef_t * pSPIx,
            uint8_t * pTxBuffer,
            uint32_t Len )
```

Sends data over SPI.

Parameters

pSPIx	Pointer to the SPI peripheral.
pTxBuffer	Pointer to the transmit buffer.
Len	Length of data to be sent.

Sends data over SPI.

Parameters

pSPIx	Pointer to the SPI peripheral.
pTxBuffer	Pointer to the transmit buffer.
Len	Length of data to be sent.

4.109.2.16 SPI_SendDataIT()

```
uint8_t SPI_SendDataIT (
            SPI_Handle_t * pSPIHandle,
            uint8_t * pTxBuffer,
            uint32_t Len )
```

Sends data over SPI using interrupt-driven communication.

Parameters

pSPIHandle	Pointer to the SPI handle structure.
pTxBuffer	Pointer to the transmit buffer.
Len	Length of data to be sent.

Note

This function sets up and initiates the transmission of data over SPI using interrupts. The data transmission will be handled asynchronously, and the user should implement the necessary interrupt handler to process data when the transmission is complete.

Returns

- SPI_READY: If the SPI is ready for communication and the data transmission is successfully initiated.
- SPI_BUSY_IN_TX: If a previous transmission is still ongoing.
- SPI_ERROR_INVALID_ARG: If the input parameters are invalid.

4.109.2.17 SPI_SSIConfig()

```
void SPI_SSIConfig (
            SPI_RegDef_t * pSPIx,
            uint8_t EnorDi )
```

Enables or disables the Software Slave Management (SSI) configuration for the SPI peripheral.

Parameters

pSPIx	Pointer to the SPI peripheral.
Enor↩ Di	ENABLE to enable SSI, DISABLE to disable SSI.

Note

SSI is used to control the slave select (NSS) pin in software. Enabling SSI ensures that the NSS pin remains high, even when the SPI is configured as a master. Disabling SSI allows the NSS pin to be controlled by the hardware or pulled to low, as specified in the SPI configuration.

4.109.2.18 SPI_SSOEConfig()

```
void SPI_SSOEConfig (
            SPI_RegDef_t * pSPIx,
            uint8_t EnorDi )
```

Enables or disables the SPI Slave Select Output Enable (SSOE) configuration for the SPI peripheral.

Parameters

pSPIx	Pointer to the SPI peripheral.
Enor↩ Di	ENABLE to enable SSOE, DISABLE to disable SSOE.

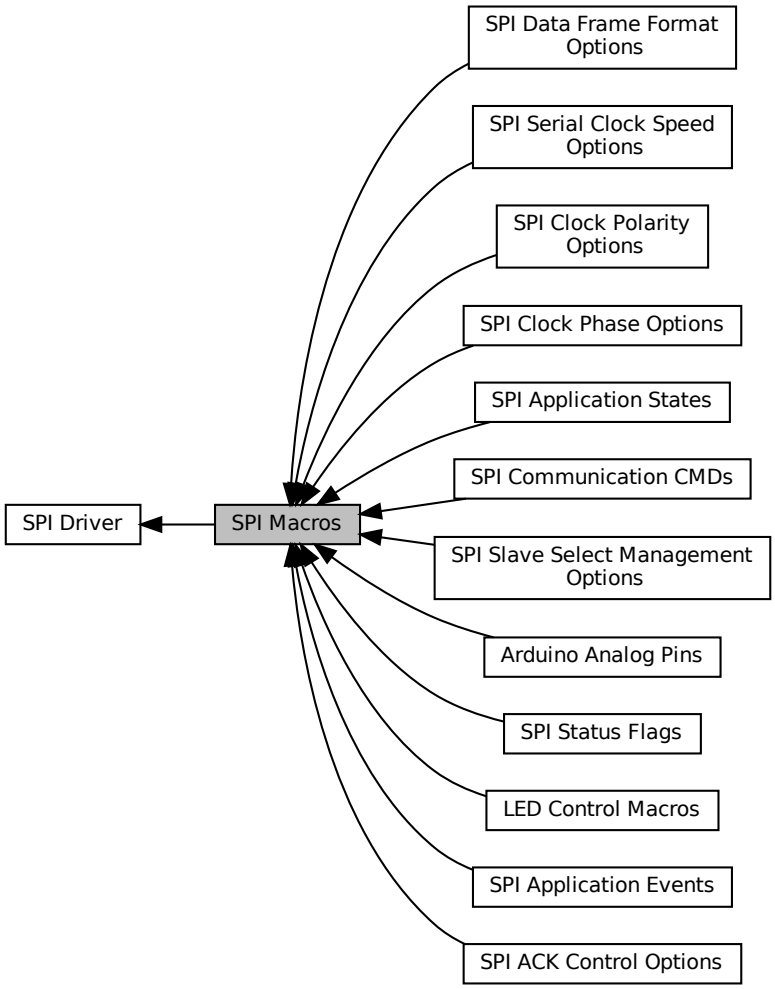
Note

SSOE is used to configure the behavior of the NSS pin when the SPI is configured as a master. When SSOE is enabled, NSS pin output is automatically managed (driven low or high) by the hardware based on the SPI state. When SSOE is disabled, NSS pin is under software control, and you must use the SPI_SSIConfig function to manage it.

4.110 SPI Macros

Macros related to SPI configuration, flags, and events.

Collaboration diagram for SPI Macros:



Modules

- SPI Application States

  Possible states of the SPI application.
- SPI ACK Control Options

  Options for controlling the ACK mechanism in SPI communication.
- SPI Serial Clock Speed Options

  Options for SPI serial clock speed.
- SPI Data Frame Format Options

  Options for SPI data frame format.
- SPI Clock Polarity Options

  Options for SPI clock polarity.
- SPI Clock Phase Options

  Options for SPI clock phase.
- SPI Slave Select Management Options

  Options for SPI slave select management.
- SPI Status Flags

  SPI Status Flags.
- SPI Application Events

  Defines possible events that can occur in the SPI application.
- SPI Communication CMDs

  Defines a set of COMMANDs for SPI communication.
- LED Control Macros

  Defines macros for controlling LEDs, including the LED state and pin number.
- Arduino Analog Pins

  Defines a set of macros representing Arduino analog pins.

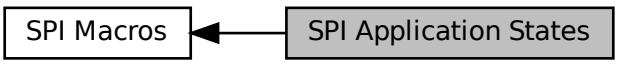
4.110.1 Detailed Description

Macros related to SPI configuration, flags, and events.

4.111 SPI Application States

Possible states of the SPI application.

Collaboration diagram for SPI Application States:



Macros

- #define SPI_DEVICE_MODE_MASTER 1
- #define SPI_DEVICE_MODE_SLAVE 0
- #define SPI_READY 0
- #define SPI_BUSY_IN_RX 1
- #define SPI_BUSY_IN_TX 2

4.111.1 Detailed Description

Possible states of the SPI application.

Defines possible states of the SPI application.

4.111.2 Macro Definition Documentation

4.111.2.1 SPI_BUSY_IN_RX

```
#define SPI_BUSY_IN_RX 1
```

SPI is busy receiving data

4.111.2.2 SPI_BUSY_IN_TX

```
#define SPI_BUSY_IN_TX 2
```

SPI is busy transmitting data

4.111.2.3 SPI_DEVICE_MODE_MASTER

```
#define SPI_DEVICE_MODE_MASTER 1
```

SPI device mode: Master

4.111.2.4 SPI_DEVICE_MODE_SLAVE

```
#define SPI_DEVICE_MODE_SLAVE 0
```

SPI device mode: Slave

4.111.2.5 SPI_READY

```
#define SPI_READY 0
```

SPI is ready for communication

4.112 SPI ACK Control Options

Options for controlling the ACK mechanism in SPI communication.

Collaboration diagram for SPI ACK Control Options:

Macros

- #define SPI_BUS_CONFIG_FULL_DUPLEX 1
- #define SPI_BUS_CONFIG_HALF_DUPLEX 2
- #define SPI_BUS_CONFIG_SIMPLEX_RX_ONLY 3

4.112.1 Detailed Description

Options for controlling the ACK mechanism in SPI communication.

4.112.2 Macro Definition Documentation

4.112.2.1 SPI_BUS_CONFIG_FULL_DUPLEX

```
#define SPI_BUS_CONFIG_FULL_DUPLEX 1
```

Full-duplex communication

4.112.2.2 SPI_BUS_CONFIG_HALF_DUPLEX

```
#define SPI_BUS_CONFIG_HALF_DUPLEX 2
```

Half-duplex communication

4.112.2.3 SPI_BUS_CONFIG_SIMPLEX_RX_ONLY

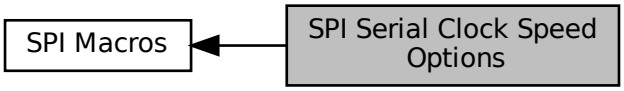
```
#define SPI_BUS_CONFIG_SIMPLEX_RX_ONLY 3
```

Simplex reception-only mode

4.113 SPI Serial Clock Speed Options

Options for SPI serial clock speed.

Collaboration diagram for SPI Serial Clock Speed Options:



Macros

- #define SPI_SCLK_SPEED_DIV2 0
- #define SPI_SCLK_SPEED_DIV4 1
- #define SPI_SCLK_SPEED_DIV8 2
- #define SPI_SCLK_SPEED_DIV16 3
- #define SPI_SCLK_SPEED_DIV32 4
- #define SPI_SCLK_SPEED_DIV64 5
- #define SPI_SCLK_SPEED_DIV128 6
- #define SPI_SCLK_SPEED_DIV256 7

4.113.1 Detailed Description

Options for SPI serial clock speed.

4.113.2 Macro Definition Documentation

4.113.2.1 SPI_SCLK_SPEED_DIV128

```
#define SPI_SCLK_SPEED_DIV128 6
```

Serial clock speed: Fpclk / 128

4.113.2.2 SPI_SCLK_SPEED_DIV16

```
#define SPI_SCLK_SPEED_DIV16 3
```

Serial clock speed: Fpclk / 16

4.113.2.3 SPI_SCLK_SPEED_DIV2

```
#define SPI_SCLK_SPEED_DIV2 0
```

Serial clock speed: Fpclk / 2

4.113.2.4 SPI_SCLK_SPEED_DIV256

```
#define SPI_SCLK_SPEED_DIV256 7
```

Serial clock speed: Fpclk / 256

4.113.2.5 SPI_SCLK_SPEED_DIV32

```
#define SPI_SCLK_SPEED_DIV32 4
```

Serial clock speed: Fpclk / 32

4.113.2.6 SPI_SCLK_SPEED_DIV4

```
#define SPI_SCLK_SPEED_DIV4 1
```

Serial clock speed: Fpclk / 4

4.113.2.7 SPI_SCLK_SPEED_DIV64

```
#define SPI_SCLK_SPEED_DIV64 5
```

Serial clock speed: Fpclk / 64

4.113.2.8 SPI_SCLK_SPEED_DIV8

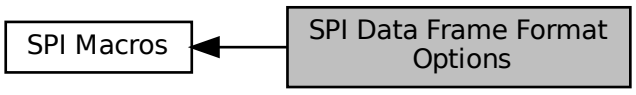
```
#define SPI_SCLK_SPEED_DIV8 2
```

Serial clock speed: Fpclk / 8

4.114 SPI Data Frame Format Options

Options for SPI data frame format.

Collaboration diagram for SPI Data Frame Format Options:



Macros

- #define SPI_DFF_8BITS 0
- #define SPI_DFF_16BITS 1

4.114.1 Detailed Description

Options for SPI data frame format.

4.114.2 Macro Definition Documentation

4.114.2.1 SPI_DFF_16BITS

```
#define SPI_DFF_16BITS 1
```

Data frame format: 16 bits per frame

4.114.2.2 SPI_DFF_8BITS

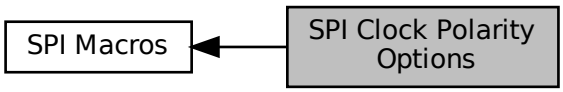
```
#define SPI_DFF_8BITS 0
```

Data frame format: 8 bits per frame

4.115 SPI Clock Polarity Options

Options for SPI clock polarity.

Collaboration diagram for SPI Clock Polarity Options:



Macros

- #define SPI_CPOL_HIGH 1
- #define SPI_CPOL_LOW 0

4.115.1 Detailed Description

Options for SPI clock polarity.

4.115.2 Macro Definition Documentation

4.115.2.1 SPI_CPOL_HIGH

```
#define SPI_CPOL_HIGH 1
```

Clock polarity: High when idle

4.115.2.2 SPI_CPOL_LOW

```
#define SPI_CPOL_LOW 0
```

Clock polarity: Low when idle

4.116 SPI Clock Phase Options

Options for SPI clock phase.

Collaboration diagram for SPI Clock Phase Options:

Macros

- #define SPI_CPHA_HIGH 1
- #define SPI_CPHA_LOW 0

4.116.1 Detailed Description

Options for SPI clock phase.

4.116.2 Macro Definition Documentation

4.116.2.1 SPI_CPHA_HIGH

```
#define SPI_CPHA_HIGH 1
```

Clock phase: Data sampled on the second edge

4.116.2.2 SPI_CPHA_LOW

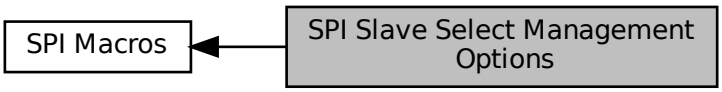
```
#define SPI_CPHA_LOW 0
```

Clock phase: Data sampled on the first edge

4.117 SPI Slave Select Management Options

Options for SPI slave select management.

Collaboration diagram for SPI Slave Select Management Options:



Macros

- #define SPI_SSM_EN 1
- #define SPI_SSM_DI 0

4.117.1 Detailed Description

Options for SPI slave select management.

4.117.2 Macro Definition Documentation

4.117.2.1 SPI_SSM_DI

```
#define SPI_SSM_DI 0
```

Slave select management: Disable (Hardware controlled)

4.117.2.2 SPI_SSM_EN

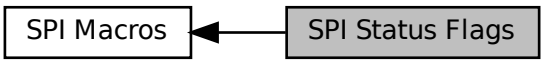
```
#define SPI_SSM_EN 1
```

Slave select management: Enable (Software controlled)

4.118 SPI Status Flags

SPI Status Flags.

Collaboration diagram for SPI Status Flags:



Macros

- #define SPI_RXNE_FLAG (1 << SPI_SR_RXNE)
- #define SPI_TXE_FLAG (1 << SPI_SR_TXE)
- #define SPI_CHSIDE_FLAG (1 << SPI_SR_CHSIDE)
- #define SPI_UDR_FLAG (1 << SPI_SR_UDR)
- #define SPI_CRCERR_FLAG (1 << SPI_SR_CRCERR)
- #define SPI_MODF_FLAG (1 << SPI_SR_MODF)
- #define SPI_OVR_FLAG (1 << SPI_SR_OVR)
- #define SPI_BUSY_FLAG (1 << SPI_SR_BSY)
- #define SPI_FRE_FLAG (1 << SPI_SR_FRE)

4.118.1 Detailed Description

SPI Status Flags.

Defines flags for SPI status.

4.118.2 Macro Definition Documentation

4.118.2.1 SPI_BUSY_FLAG

```
#define SPI_BUSY_FLAG (1 << SPI_SR_BSY)
```

Busy flag

4.118.2.2 SPI_CHSIDE_FLAG

```
#define SPI_CHSIDE_FLAG (1 << SPI_SR_CHSIDE)
```

Channel side flag

4.118.2.3 SPI_CRCERR_FLAG

```
#define SPI_CRCERR_FLAG (1 << SPI_SR_CRCERR)
```

CRC error flag

4.118.2.4 SPI_FRE_FLAG

```
#define SPI_FRE_FLAG (1 << SPI_SR_FRE)
```

Frame format error flag

4.118.2.5 SPI_MODF_FLAG

```
#define SPI_MODF_FLAG (1 << SPI_SR_MODF)
```

Mode fault flag

4.118.2.6 SPI_OVR_FLAG

```
#define SPI_OVR_FLAG (1 << SPI_SR_OVR)
```

Overrun flag

4.118.2.7 SPI_RXNE_FLAG

```
#define SPI_RXNE_FLAG (1 << SPI_SR_RXNE)
```

Receive buffer not empty flag

4.118.2.8 SPI_TXE_FLAG

```
#define SPI_TXE_FLAG (1 << SPI_SR_TXE)
```

Transmit buffer empty flag

4.118.2.9 SPI_UDR_FLAG

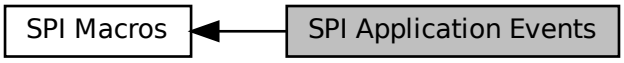
```
#define SPI_UDR_FLAG (1 << SPI_SR_UDR)
```

Underrun flag

4.119 SPI Application Events

Defines possible events that can occur in the SPI application.

Collaboration diagram for SPI Application Events:



Macros

- #define SPI_EVENT_TX_CMPLT 1
- #define SPI_EVENT_RX_CMPLT 2
- #define SPI_EVENT_OVR_ERR 3
- #define SPI_EVENT_CRC_ERR 4

4.119.1 Detailed Description

Defines possible events that can occur in the SPI application.

4.119.2 Macro Definition Documentation

4.119.2.1 SPI_EVENT_CRC_ERR

```
#define SPI_EVENT_CRC_ERR 4
```

Event indicating a CRC (Cyclic Redundancy Check) error in SPI communication.

4.119.2.2 SPI_EVENT_OVR_ERR

```
#define SPI_EVENT_OVR_ERR 3
```

Event indicating an overrun error during SPI communication.

4.119.2.3 SPI_EVENT_RX_CMPLT

```
#define SPI_EVENT_RX_CMPLT 2
```

Event indicating the completion of a receive operation.

4.119.2.4 SPI_EVENT_TX_CMPLT

```
#define SPI_EVENT_TX_CMPLT 1
```

Event indicating the completion of a transmit operation.

4.120 SPI Communication CMDs

Defines a set of COMMANDs for SPI communication.

Collaboration diagram for SPI Communication CMDs:

Macros

- #define CMD_LED_CTRL 0x50
- #define CMD_SENSOR_READ 0x51
- #define CMD_LED_READ 0x52
- #define CMD_PRINT 0x53
- #define CMD_ID_READ 0x54

4.120.1 Detailed Description

Defines a set of COMMANDs for SPI communication.

4.120.2 Macro Definition Documentation

4.120.2.1 CMD_ID_READ

```
#define CMD_ID_READ 0x54
```

CMD to read an ID.

4.120.2.2 CMD_LED_CTRL

```
#define CMD_LED_CTRL 0x50
```

CMD to control an LED.

4.120.2.3 CMD_LED_READ

```
#define CMD_LED_READ 0x52
```

CMD to read the LED state.

4.120.2.4 CMD_PRINT

```
#define CMD_PRINT 0x53
```

CMD to print data.

4.120.2.5 CMD_SENSOR_READ

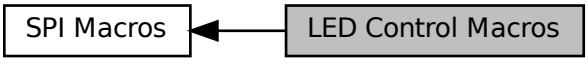
```
#define CMD_SENSOR_READ 0x51
```

CMD to read a sensor.

4.121 LED Control Macros

Defines macros for controlling LEDs, including the LED state and pin number.

Collaboration diagram for LED Control Macros:



Macros

- #define LED_ON 1
- #define LED_OFF 0
- #define LED_PIN 9

4.121.1 Detailed Description

Defines macros for controlling LEDs, including the LED state and pin number.

4.121.2 Macro Definition Documentation

4.121.2.1 LED_OFF

```
#define LED_OFF 0
```

Represents the LED OFF state.

4.121.2.2 LED_ON

```
#define LED_ON 1
```

Represents the LED ON state.

4.121.2.3 LED_PIN

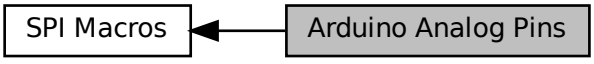
```
#define LED_PIN 9
```

Represents the Arduino LED pin number.

4.122 Arduino Analog Pins

Defines a set of macros representing Arduino analog pins.

Collaboration diagram for Arduino Analog Pins:



Macros

- #define ANALOG_PIN0 0
- #define ANALOG_PIN1 1
- #define ANALOG_PIN2 2
- #define ANALOG_PIN3 3
- #define ANALOG_PIN4 4

4.122.1 Detailed Description

Defines a set of macros representing Arduino analog pins.

4.122.2 Macro Definition Documentation

4.122.2.1 ANALOG_PIN0

```
#define ANALOG_PIN0 0
```

Represents analog pin 0.

4.122.2.2 ANALOG_PIN1

```
#define ANALOG_PIN1 1
```

Represents analog pin 1.

4.122.2.3 ANALOG_PIN2

```
#define ANALOG_PIN2 2
```

Represents analog pin 2.

4.122.2.4 ANALOG_PIN3

```
#define ANALOG_PIN3 3
```

Represents analog pin 3.

4.122.2.5 ANALOG_PIN4

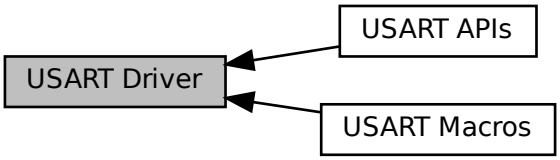
```
#define ANALOG_PIN4 4
```

Represents analog pin 4.

4.123 USART Driver

USART driver APIs for STM32F407xx MCU.

Collaboration diagram for USART Driver:



Modules

- USART APIs

  USART driver APIs for STM32F407xx MCU.
- USART Macros

  Macros related to USART configuration, flags, and events.

Classes

- struct USART_Config_t

  Configuration structure for USART (Universal Synchronous Asynchronous Receiver Transmitter) peripheral.
- struct USART_Handle_t

  Handle structure for USART peripheral.

4.123.1 Detailed Description

USART driver APIs for STM32F407xx MCU.

4.124 USART APIs

USART driver APIs for STM32F407xx MCU.

Collaboration diagram for USART APIs:

Functions

- void USART_PeripheralClockControl (USART_RegDef_t *pUSARTx, uint8_t EnorDi)
  Enable or disable the peripheral clock for the given USARTx.
- void USART_SetBaudRate (USART_RegDef_t *pUSARTx, uint32_t BaudRate)
  Set the baud rate for a USART peripheral.
- void USART_Init (USART_Handle_t *pUSARTHandle)
  Initialize the USART peripheral.
- void USART_DeInit (USART_RegDef_t *pUSARTx)
  Deinitialize the USART peripheral.
- void USART_SendData (USART_Handle_t *pUSARTHandle, uint8_t *pTxBuffer, uint32_t Len)
  Send data over USART.
- void USART_ReceiveData (USART_Handle_t *pUSARTHandle, uint8_t *pRxBuffer, uint32_t Len)
  Receive data from USART.
- uint8_t USART_SendDataIT (USART_Handle_t *pUSARTHandle, uint8_t *pTxBuffer, uint32_t Len)
  Send data over USART using interrupt-driven communication.
- uint8_t USART_ReceiveDataIT (USART_Handle_t *pUSARTHandle, uint8_t *pRxBuffer, uint32_t Len)
  Receive data from USART using interrupt-driven communication.
- void USART_PeripheralControl (USART_RegDef_t *pUSARTx, uint8_t EnorDi)
  Control the USART peripheral (ENABLE/DISABLE).
- uint8_t USART_GetFlagStatus (USART_RegDef_t *pUSARTx, uint8_t FlagName)
  Get the status of a specific USART flag.
- void USART_ClearFlag (USART_RegDef_t *pUSARTx, uint16_t FlagName)
  Clear a specific USART flag.
- void USART_CloseTransmission (USART_Handle_t *pUSARTHandle)
  Close USART transmission.
- void USART_CloseReception (USART_Handle_t *pUSARTHandle)
  Close USART reception.
- void USART_IRQInterruptConfig (uint8_t IRQNumber, uint8_t EnorDi)
  Configure IRQ number and enable/disable IRQ.
- void USART_IRQPriorityConfig (uint8_t IRQNumber, uint32_t IRQPriority)
  Set the priority of an IRQ.
- void USART_IRQHandling (USART_Handle_t *pUSARTHandle)
  Handle USART interrupts.
- void USART_ApplicationEventCallback (USART_Handle_t *pUSARTHandle, uint8_t AppEv)
  Application callback function for USART events.
- void USART_ClearEventErrFlag (USART_RegDef_t *pUSARTx)
  Clears the error flags in the USART status register.

4.124.1 Detailed Description

USART driver APIs for STM32F407xx MCU.

4.124.2 Function Documentation

4.124.2.1 USART_ApplicationEventCallback()

```
void USART_ApplicationEventCallback (
            USART_Handle_t * pUSARTHandle,
            uint8_t AppEv )
```

Application callback function for USART events.

Parameters

pUSARTHandle	Pointer to the USART handle structure.
AppEv	USART application event.

This function serves as a callback that can be overridden by the application to handle USART-related events. It is called when specific events occur during USART communication and allows the application to take custom actions.

Parameters

pUSARTHandle	Pointer to the USART handle structure.
AppEv	USART application event.

Note

This is a weak implementation, and the application can override this function to provide custom event handling.

4.124.2.2 USART_ClearEventErrFlag()

```
void USART_ClearEventErrFlag (
            USART_RegDef_t * pUSARTx )
```

Clears the error flags in the USART status register.

This function clears any error flags that may have been set in the USART status register (SR). It is commonly used to clear error flags before resuming USART communication after an error condition.

Parameters

pUSARTx	Pointer to the USART peripheral.

Warning

Do not call this function during active USART communication, as it may cause data loss.

Clears the error flags in the USART status register.

This function clears any error flags that may have been set in the USART status register (SR). It is commonly used to clear error flags before resuming USART communication after an error condition.

Parameters

pUSARTx	Pointer to the USART peripheral.

Warning

Do not call this function during active USART communication, as it may cause data loss.

4.124.2.3 USART_ClearFlag()

```
void USART_ClearFlag (
            USART_RegDef_t * pUSARTx,
            uint16_t FlagName )
```

Clear a specific USART flag.

Parameters

pUSARTx	Pointer to the USART peripheral.
FlagName	Name of the flag to clear.

4.124.2.4 USART_CloseReception()

```
void USART_CloseReception (
            USART_Handle_t * pUSARTHandle )
```

Close USART reception.

Parameters

pUSARTHandle	Pointer to the USART handle structure.

4.124.2.5 USART_CloseTransmission()

```
void USART_CloseTransmission (
            USART_Handle_t * pUSARTHandle )
```

Close USART transmission.

Parameters

pUSARTHandle	Pointer to the USART handle structure.

4.124.2.6 USART_DeInit()

```
void USART_DeInit (
            USART_RegDef_t * pUSARTx )
```

Deinitialize the USART peripheral.

Parameters

pUSARTx	Pointer to the USART peripheral.

4.124.2.7 USART_GetFlagStatus()

```
uint8_t USART_GetFlagStatus (
            USART_RegDef_t * pUSARTx,
            uint8_t FlagName )
```

Get the status of a specific USART flag.

Parameters

pUSARTx	Pointer to the USART peripheral.
FlagName	Name of the flag to check.

Returns

uint8_t Status of the flag (FLAG_SET or FLAG_RESET).

4.124.2.8 USART_Init()

```
void USART_Init (
            USART_Handle_t * pUSARTHandle )
```

Initialize the USART peripheral.

Parameters

pUSARTHandle	Pointer to the USART handle structure.

4.124.2.9 USART_IRQHandling()

```
void USART_IRQHandling (
```

```
            USART_Handle_t * pUSARTHandle )
```

Handle USART interrupts.

Parameters

pUSARTHandle	Pointer to the USART handle structure.

4.124.2.10 USART_IRQInterruptConfig()

```
void USART_IRQInterruptConfig (
            uint8_t IRQNumber,
            uint8_t EnorDi )
```

Configure IRQ number and enable/disable IRQ.

Parameters

IRQNumber	IRQ number to configure.
EnorDi	ENABLE or DISABLE IRQ.

4.124.2.11 USART_IRQPriorityConfig()

```
void USART_IRQPriorityConfig (
            uint8_t IRQNumber,
            uint32_t IRQPriority )
```

Set the priority of an IRQ.

Parameters

IRQNumber	IRQ number to set priority for.
IRQPriority	Priority to be set.

4.124.2.12 USART_PeripheralClockControl()

```
void USART_PeripheralClockControl (
            USART_RegDef_t * pUSARTx,
            uint8_t EnorDi )
```

Enable or disable the peripheral clock for the given USARTx.

Parameters

pUSARTx	Pointer to the USART peripheral.
EnorDi	ENABLE or DISABLE the clock.

4.124.2.13 USART_PeripheralControl()

```
void USART_PeripheralControl (
             USART_RegDef_t * pUSARTx,
             uint8_t EnorDi )
```

Control the USART peripheral (ENABLE/DISABLE).

Parameters

pUSARTx	Pointer to the USART peripheral.
EnorDi	ENABLE or DISABLE the peripheral.

4.124.2.14 USART_ReceiveData()

```
void USART_ReceiveData (
             USART_Handle_t * pUSARTHandle,
             uint8_t * pRxBuffer,
             uint32_t Len )
```

Receive data from USART.

Parameters

pUSARTHandle	Pointer to the USART handle structure.
pRxBuffer	Pointer to the receive buffer.
Len	Length of data to be received.

4.124.2.15 USART_ReceiveDataIT()

```
uint8_t USART_ReceiveDataIT (
             USART_Handle_t * pUSARTHandle,
             uint8_t * pRxBuffer,
             uint32_t Len )
```

Receive data from USART using interrupt-driven communication.

Parameters

pUSARTHandle	Pointer to the USART handle structure.
pRxBuffer	Pointer to the receive buffer.
Len	Length of data to be received.

Returns

uint8_t Reception status.

4.124.2.16 USART_SendData()

```
void USART_SendData (
             USART_Handle_t * pUSARTHandle,
             uint8_t * pTxBuffer,
             uint32_t Len )
```

Send data over USART.

Parameters

pUSARTHandle	Pointer to the USART handle structure.
pTxBuffer	Pointer to the transmit buffer.
Len	Length of data to be sent.

4.124.2.17 USART_SendDataIT()

```
uint8_t USART_SendDataIT (
             USART_Handle_t * pUSARTHandle,
             uint8_t * pTxBuffer,
             uint32_t Len )
```

Send data over USART using interrupt-driven communication.

Parameters

pUSARTHandle	Pointer to the USART handle structure.
pTxBuffer	Pointer to the transmit buffer.
Len	Length of data to be sent.

Returns

uint8_t Transmission status.

4.124.2.18 USART_SetBaudRate()

```
void USART_SetBaudRate (
             USART_RegDef_t * pUSARTx,
             uint32_t BaudRate )
```

Set the baud rate for a USART peripheral.

This function calculates and sets the baud rate for a USART peripheral based on the provided baud rate and the system's APB clock frequency.

Parameters

pUSARTx	Pointer to the USART peripheral.
BaudRate	Desired baud rate (in bits per second).

Note

The function assumes that the USART is already initialized and configured.

Warning

This function may not work as expected if the USART peripheral is not properly configured.

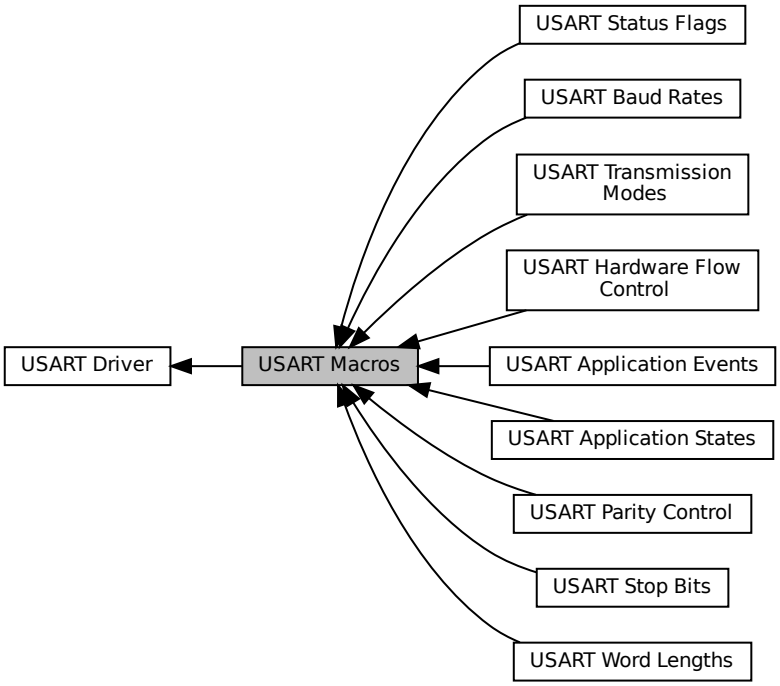
Returns

None

4.125 USART Macros

Macros related to USART configuration, flags, and events.

Collaboration diagram for USART Macros:



Modules

- USART Transmission Modes

  Possible options for USART transmission mode.
- USART Baud Rates

  Possible options for USART baud rates.
- USART Parity Control

  Possible options for USART parity control.
- USART Word Lengths

  Possible options for USART word length.
- USART Stop Bits

  Possible options for USART stop bits.
- USART Hardware Flow Control

  Possible options for USART hardware flow control.
- USART Status Flags

  USART Status Flags.
- USART Application States

  Defines possible states of the USART application.
- USART Application Events

  Defines possible events that can occur in the USART application.

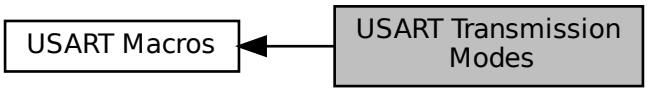
4.125.1 Detailed Description

Macros related to USART configuration, flags, and events.

4.126 USART Transmission Modes

Possible options for USART transmission mode.

Collaboration diagram for USART Transmission Modes:



Macros

- #define USART_MODE_ONLY_TX 0
- #define USART_MODE_ONLY_RX 1
- #define USART_MODE_TXRX 2

4.126.1 Detailed Description

Possible options for USART transmission mode.

4.126.2 Macro Definition Documentation

4.126.2.1 USART_MODE_ONLY_RX

```
#define USART_MODE_ONLY_RX 1
```

Only receive mode.

4.126.2.2 USART_MODE_ONLY_TX

```
#define USART_MODE_ONLY_TX 0
```

Only transmit mode.

4.126.2.3 USART_MODE_TXRX

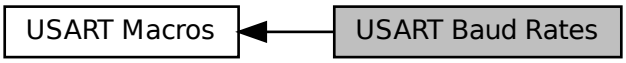
```
#define USART_MODE_TXRX 2
```

Transmit and receive mode.

4.127 USART Baud Rates

Possible options for USART baud rates.

Collaboration diagram for USART Baud Rates:



Macros

- #define **USART_STD_BAUD_1200** (uint8_t)1200
- #define **USART_STD_BAUD_2400** (uint8_t)2400
- #define **USART_STD_BAUD_9600** (uint8_t)9600
- #define **USART_STD_BAUD_19200** (uint8_t)19200
- #define **USART_STD_BAUD_38400** (uint8_t)38400
- #define **USART_STD_BAUD_57600** (uint8_t)57600
- #define **USART_STD_BAUD_115200** (uint8_t)115200
- #define **USART_STD_BAUD_230400** (uint8_t)230400
- #define **USART_STD_BAUD_460800** (uint8_t)460800
- #define **USART_STD_BAUD_921600** (uint8_t)921600
- #define **USART_STD_BAUD_2M** (uint8_t)2000000
- #define **USART_STD_BAUD_3M** (uint8_t)3000000

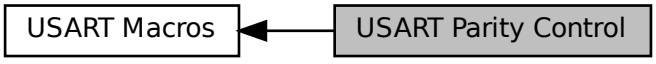
4.127.1 Detailed Description

Possible options for USART baud rates.

4.128 USART Parity Control

Possible options for USART parity control.

Collaboration diagram for USART Parity Control:



Macros

- #define USART_PARITY_EN_ODD 2
- #define USART_PARITY_EN_EVEN 1
- #define USART_PARITY_DISABLE 0

4.128.1 Detailed Description

Possible options for USART parity control.

4.128.2 Macro Definition Documentation

4.128.2.1 USART_PARITY_DISABLE

```
#define USART_PARITY_DISABLE 0
```

Disable USART parity control.

4.128.2.2 USART_PARITY_EN_EVEN

```
#define USART_PARITY_EN_EVEN 1
```

Enable USART parity control with even parity.

4.128.2.3 USART_PARITY_EN_ODD

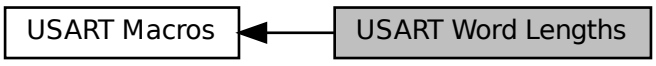
```
#define USART_PARITY_EN_ODD 2
```

Enable USART parity control with odd parity.

4.129 USART Word Lengths

Possible options for USART word length.

Collaboration diagram for USART Word Lengths:



Macros

- #define USART_WORDLEN_8BITS 0
- #define USART_WORDLEN_9BITS 1

4.129.1 Detailed Description

Possible options for USART word length.

4.129.2 Macro Definition Documentation

4.129.2.1 USART_WORDLEN_8BITS

```
#define USART_WORDLEN_8BITS 0
```

8-bit data word length.

4.129.2.2 USART_WORDLEN_9BITS

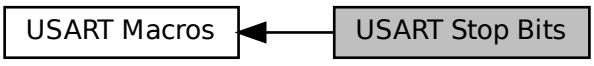
```
#define USART_WORDLEN_9BITS 1
```

9-bit data word length.

4.130 USART Stop Bits

Possible options for USART stop bits.

Collaboration diagram for USART Stop Bits:



Macros

- #define USART_STOPBITS_1 0
- #define USART_STOPBITS_0_5 1
- #define USART_STOPBITS_2 2
- #define USART_STOPBITS_1_5 3

4.130.1 Detailed Description

Possible options for USART stop bits.

4.130.2 Macro Definition Documentation

4.130.2.1 USART_STOPBITS_0_5

```
#define USART_STOPBITS_0_5 1
```

0.5 stop bits.

4.130.2.2 USART_STOPBITS_1

```
#define USART_STOPBITS_1 0
```

1 stop bit.

4.130.2.3 USART_STOPBITS_1_5

```
#define USART_STOPBITS_1_5 3
```

1.5 stop bits.

4.130.2.4 USART_STOPBITS_2

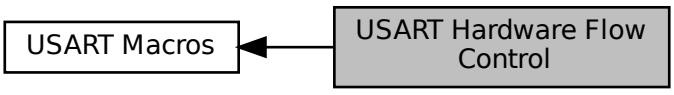
```
#define USART_STOPBITS_2 2
```

2 stop bits.

4.131 USART Hardware Flow Control

Possible options for USART hardware flow control.

Collaboration diagram for USART Hardware Flow Control:



Macros

- #define USART_HW_FLOW_CTRL_NONE 0
- #define USART_HW_FLOW_CTRL_CTS 1
- #define USART_HW_FLOW_CTRL_RTS 2
- #define USART_HW_FLOW_CTRL_CTS_RTS 3

4.131.1 Detailed Description

Possible options for USART hardware flow control.

4.131.2 Macro Definition Documentation

4.131.2.1 USART_HW_FLOW_CTRL_CTS

```
#define USART_HW_FLOW_CTRL_CTS 1
```

Hardware flow control using CTS (Clear To Send).

4.131.2.2 USART_HW_FLOW_CTRL_CTS_RTS

```
#define USART_HW_FLOW_CTRL_CTS_RTS 3
```

Hardware flow control using both CTS and RTS.

4.131.2.3 USART_HW_FLOW_CTRL_NONE

```
#define USART_HW_FLOW_CTRL_NONE 0
```

No hardware flow control.

4.131.2.4 USART_HW_FLOW_CTRL_RTS

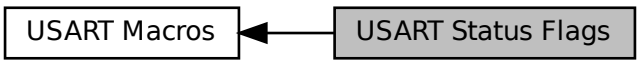
```
#define USART_HW_FLOW_CTRL_RTS 2
```

Hardware flow control using RTS (Request To Send).

4.132 USART Status Flags

USART Status Flags.

Collaboration diagram for USART Status Flags:



Macros

- #define USART_FLAG_PE (1 << USART_SR_PE)
- #define USART_FLAG_FE (1 << USART_SR_FE)
- #define USART_FLAG_NE (1 << USART_SR_NE)
- #define USART_FLAG_ORE (1 << USART_SR_ORE)
- #define USART_FLAG_IDLE (1 << USART_SR_IDLE)
- #define USART_FLAG_RXNE (1 << USART_SR_RXNE)
- #define USART_FLAG_TC (1 << USART_SR_TC)
- #define USART_FLAG_TXE (1 << USART_SR_TXE)

4.132.1 Detailed Description

USART Status Flags.

Defines flags for USART status.

4.132.2 Macro Definition Documentation

4.132.2.1 USART_FLAG_FE

```
#define USART_FLAG_FE (1 << USART_SR_FE)
```

USART Framing error flag

4.132.2.2 USART_FLAG_IDLE

```
#define USART_FLAG_IDLE (1 << USART_SR_IDLE)
```

USART Idle line detected flag

4.132.2.3 USART_FLAG_NE

```
#define USART_FLAG_NE (1 << USART_SR_NE)
```

USART Noise error flag

4.132.2.4 USART_FLAG_ORE

```
#define USART_FLAG_ORE (1 << USART_SR_ORE)
```

USART Overrun error flag

4.132.2.5 USART_FLAG_PE

```
#define USART_FLAG_PE (1 << USART_SR_PE)
```

USART Parity error flag

4.132.2.6 USART_FLAG_RXNE

```
#define USART_FLAG_RXNE (1 << USART_SR_RXNE)
```

USART Receive buffer not empty flag

4.132.2.7 USART_FLAG_TC

```
#define USART_FLAG_TC (1 << USART_SR_TC)
```

USART Transmission complete flag

4.132.2.8 USART_FLAG_TXE

```
#define USART_FLAG_TXE (1 << USART_SR_TXE)
```

USART Transmit buffer empty flag

4.133 USART Application States

Defines possible states of the USART application.

Collaboration diagram for USART Application States:

Macros

- #define USART_READY 0
- #define USART_BUSY_IN_RX 1
- #define USART_BUSY_IN_TX 2

4.133.1 Detailed Description

Defines possible states of the USART application.

4.133.2 Macro Definition Documentation

4.133.2.1 USART_BUSY_IN_RX

```
#define USART_BUSY_IN_RX 1
```

USART is busy receiving data

4.133.2.2 USART_BUSY_IN_TX

```
#define USART_BUSY_IN_TX 2
```

USART is busy transmitting data

4.133.2.3 USART_READY

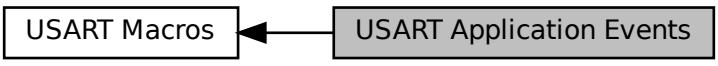
```
#define USART_READY 0
```

USART is ready for communication

4.134 USART Application Events

Defines possible events that can occur in the USART application.

Collaboration diagram for USART Application Events:



Macros

- #define USART_EVENT_TX_CMPLT 0
- #define USART_EVENT_RX_CMPLT 1
- #define USART_EVENT_IDLE 2
- #define USART_EVENT_CTS 3
- #define USART_EVENT_PE 4
- #define USART_ERR_FE 5
- #define USART_ERR_NE 6
- #define USART_ERR_ORE 7

4.134.1 Detailed Description

Defines possible events that can occur in the USART application.

4.134.2 Macro Definition Documentation

4.134.2.1 USART_ERR_FE

```
#define USART_ERR_FE 5
```

Interrupt Event indicating a framing error during USART communication.

4.134.2.2 USART_ERR_NE

```
#define USART_ERR_NE 6
```

Interrupt Event indicating a noise error during USART communication.

4.134.2.3 USART_ERR_ORE

```
#define USART_ERR_ORE 7
```

Interrupt Event indicating an overrun error during USART communication.

4.134.2.4 USART_EVENT_CTS

```
#define USART_EVENT_CTS 3
```

Interrupt Event indicating a change in CTS (Clear To Send) signal during USART communication.

4.134.2.5 USART_EVENT_IDLE

```
#define USART_EVENT_IDLE 2
```

Interrupt Event indicating an idle line detection during USART communication.

4.134.2.6 USART_EVENT_PE

```
#define USART_EVENT_PE 4
```

Interrupt Event indicating a parity error during USART communication.

4.134.2.7 USART_EVENT_RX_CMPLT

```
#define USART_EVENT_RX_CMPLT 1
```

Interrupt Event indicating the completion of a receive operation.

4.134.2.8 USART_EVENT_TX_CMPLT

```
#define USART_EVENT_TX_CMPLT 0
```

Interrupt Event indicating the completion of a transmit operation.

4.135 SPI Private Helper Functions

These are private helper functions for managing interrupts in SPI communication.

4.135.1 Detailed Description

These are private helper functions for managing interrupts in SPI communication.

4.136 USART Private Helper Functions

These are private helper functions for managing interrupts in USART communication.

4.136.1 Detailed Description

These are private helper functions for managing interrupts in USART communication.

Chapter 5

Class Documentation

5.1 EXTI_RegDef_t Struct Reference

EXTI peripheral register definition structure.

`#include <stm32f407xx.h>`

Public Attributes

- __vo uint32_t IMR
- __vo uint32_t EMR
- __vo uint32_t RTSR
- __vo uint32_t FTSR
- __vo uint32_t SWIER
- __vo uint32_t PR

5.1.1 Detailed Description

EXTI peripheral register definition structure.

5.1.2 Member Data Documentation

5.1.2.1 EMR

`__vo uint32_t EXTI_RegDef_t::EMR`

EXTI event mask register

5.1.2.2 FTSR

```
__vo uint32_t EXTI_RegDef_t::FTSR
```

EXTI falling trigger selection register

5.1.2.3 IMR

```
__vo uint32_t EXTI_RegDef_t::IMR
```

EXTI interrupt mask register

5.1.2.4 PR

```
__vo uint32_t EXTI_RegDef_t::PR
```

EXTI pending register

5.1.2.5 RTSR

```
__vo uint32_t EXTI_RegDef_t::RTSR
```

EXTI rising trigger selection register

5.1.2.6 SWIER

```
__vo uint32_t EXTI_RegDef_t::SWIER
```

EXTI software interrupt event register

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

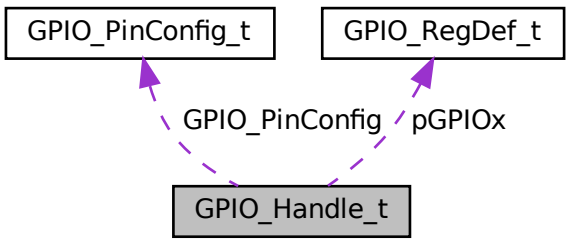
- drivers/Inc/stm32f407xx.h

5.2 GPIO_Handle_t Struct Reference

Handle structure for GPIO pin.

```
#include <stm32f407xx_gpio_driver.h>
```

Collaboration diagram for GPIO_Handle_t:



Public Attributes

- GPIO_RegDef_t * pGPIOx
- GPIO_PinConfig_t GPIO_PinConfig

5.2.1 Detailed Description

Handle structure for GPIO pin.

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

- drivers/Inc/stm32f407xx_gpio_driver.h

5.3 GPIO_PinConfig_t Struct Reference

Configuration structure for GPIO pin.

```
#include <stm32f407xx_gpio_driver.h>
```

Public Attributes

- uint32_t GPIO_PinNumber
- uint32_t GPIO_PinMode
- uint32_t GPIO_PinSpeed
- uint32_t GPIO_PinPuPdControl
- uint32_t GPIO_PinPinOPType
- uint32_t GPIO_PinAltFunMode

5.3.1 Detailed Description

Configuration structure for GPIO pin.

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

- drivers/Inc/stm32f407xx_gpio_driver.h

5.4 GPIO_RegDef_t Struct Reference

GPIO peripheral register definition structure.

```
#include <stm32f407xx.h>
```

Public Attributes

- __vo uint32_t MODER
- __vo uint32_t OTYPER
- __vo uint32_t OSPEEDR
- __vo uint32_t PUPDR
- __vo uint32_t IDR
- __vo uint32_t ODR
- __vo uint32_t BSRR
- __vo uint32_t LCKR
- __vo uint32_t AFR [2]

5.4.1 Detailed Description

GPIO peripheral register definition structure.

5.4.2 Member Data Documentation

5.4.2.1 AFR

```
__vo uint32_t GPIO_RegDef_t::AFR[2]
```

GPIO alternate function registers (AFR[0] = low, AFR[1] = high)

5.4.2.2 BSRR

```
__vo uint32_t GPIO_RegDef_t::BSRR
```

GPIO port bit set/reset register

5.4.2.3 IDR

```
__vo uint32_t GPIO_RegDef_t::IDR
```

GPIO port input data register

5.4.2.4 LCKR

```
__vo uint32_t GPIO_RegDef_t::LCKR
```

GPIO port configuration lock register

5.4.2.5 MODER

```
__vo uint32_t GPIO_RegDef_t::MODER
```

GPIO port mode register

5.4.2.6 ODR

```
__vo uint32_t GPIO_RegDef_t::ODR
```

GPIO port output data register

5.4.2.7 OSPEEDR

```
__vo uint32_t GPIO_RegDef_t::OSPEEDR
```

GPIO port output speed register

5.4.2.8 OTYPER

```
__vo uint32_t GPIO_RegDef_t::OTYPER
```

GPIO port output type register

5.4.2.9 PUPDR

```
__vo uint32_t GPIO_RegDef_t::PUPDR
```

GPIO port pull-up/pull-down register

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

- drivers/Inc/stm32f407xx.h

5.5 I2C_Config_t Struct Reference

Configuration structure for I2C peripheral.

```
#include <stm32f407xx_i2c_driver.h>
```

Public Attributes

- uint32_t **I2C_SCLspeed**
- uint8_t **I2C_DeviceAdress**
- uint8_t **I2C_ACKControl**
- uint8_t **I2C_FMDutyCycle**

5.5.1 Detailed Description

Configuration structure for I2C peripheral.

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

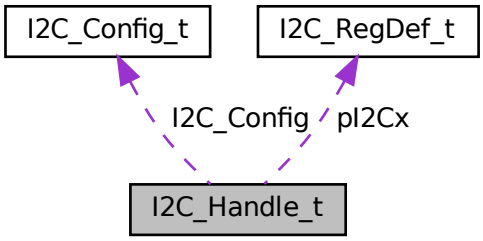
- drivers/Inc/stm32f407xx_i2c_driver.h

5.6 I2C_Handle_t Struct Reference

Handle structure for I2C peripheral.

`#include <stm32f407xx_i2c_driver.h>`

Collaboration diagram for I2C_Handle_t:



Public Attributes

- I2C_RegDef_t * **pI2Cx**
- I2C_Config_t **I2C_Config**
- uint8_t * **pTxBuffer**
- uint8_t * **pRxBuffer**
- uint32_t **TxLen**
- uint32_t **RxLen**
- uint8_t **TxRxState**
- uint8_t **DevAddr**
- uint32_t **RxSize**
- uint8_t **Sr**

5.6.1 Detailed Description

Handle structure for I2C peripheral.

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

- drivers/Inc/stm32f407xx_i2c_driver.h

5.7 I2C_RegDef_t Struct Reference

I2C peripheral register definition structure.

```
#include <stm32f407xx.h>
```

Public Attributes

- __vo uint32_t CR1
- __vo uint32_t CR2
- __vo uint32_t OAR1
- __vo uint32_t OAR2
- __vo uint32_t DR
- __vo uint32_t SR1
- __vo uint32_t SR2
- __vo uint32_t CCR
- __vo uint32_t TRISE
- __vo uint32_t FLTR

5.7.1 Detailed Description

I2C peripheral register definition structure.

5.7.2 Member Data Documentation

5.7.2.1 CCR

```
__vo uint32_t I2C_RegDef_t::CCR
```

I2C clock control register

5.7.2.2 CR1

```
__vo uint32_t I2C_RegDef_t::CR1
```

I2C control register 1

5.7.2.3 CR2

```
__vo uint32_t I2C_RegDef_t::CR2
```

I2C control register 2

5.7.2.4 DR

```
__vo uint32_t I2C_RegDef_t::DR
```

I2C data register

5.7.2.5 FLTR

```
__vo uint32_t I2C_RegDef_t::FLTR
```

I2C digital filter register

5.7.2.6 OAR1

```
__vo uint32_t I2C_RegDef_t::OAR1
```

I2C own address register 1

5.7.2.7 OAR2

```
__vo uint32_t I2C_RegDef_t::OAR2
```

I2C own address register 2

5.7.2.8 SR1

```
__vo uint32_t I2C_RegDef_t::SR1
```

I2C status register 1

5.7.2.9 SR2

```
__vo uint32_t I2C_RegDef_t::SR2
```

I2C status register 2

5.7.2.10 TRISE

```
__vo uint32_t I2C_RegDef_t::TRISE
```

I2C rise time register

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

- drivers/Inc/stm32f407xx.h

5.8 RCC_RegDef_t Struct Reference

RCC peripheral register definition structure.

```
#include <stm32f407xx.h>
```

Public Attributes

- __vo uint32_t CR
- __vo uint32_t PLLCFGR
- __vo uint32_t CFGR
- __vo uint32_t CIR
- __vo uint32_t AHB1RSTR
- __vo uint32_t AHB2RSTR
- __vo uint32_t AHB3RSTR
- uint32_t **RESERVED0**
- __vo uint32_t APB1RSTR
- __vo uint32_t APB2RSTR
- uint32_t **RESERVED1** [2]
- __vo uint32_t AHB1ENR
- __vo uint32_t AHB2ENR
- __vo uint32_t AHB3ENR
- uint32_t **RESERVED2**
- __vo uint32_t APB1ENR
- __vo uint32_t APB2ENR
- uint32_t **RESERVED3** [2]
- __vo uint32_t AHB1LPENR
- __vo uint32_t AHB2LPENR
- __vo uint32_t AHB3LPENR
- uint32_t **RESERVED4**
- __vo uint32_t APB1LPENR
- __vo uint32_t APB2LPENR
- uint32_t **RESERVED5** [2]
- __vo uint32_t BDCR
- __vo uint32_t CSR
- uint32_t **RESERVED6** [2]
- __vo uint32_t SSCGR
- __vo uint32_t PLLI2SCFGR
- __vo uint32_t PLLSAICFGR
- __vo uint32_t DCKCFGR
- __vo uint32_t CKGATENR
- __vo uint32_t DCKCFGR2

5.8.1 Detailed Description

RCC peripheral register definition structure.

5.8.2 Member Data Documentation

5.8.2.1 AHB1ENR

```
__vo uint32_t RCC_RegDef_t::AHB1ENR
```

RCC AHB1 peripheral clock enable register

5.8.2.2 AHB1LPENR

```
__vo uint32_t RCC_RegDef_t::AHB1LPENR
```

RCC AHB1 peripheral clock enable in low power mode register

5.8.2.3 AHB1RSTR

```
__vo uint32_t RCC_RegDef_t::AHB1RSTR
```

RCC AHB1 peripheral reset register

5.8.2.4 AHB2ENR

```
__vo uint32_t RCC_RegDef_t::AHB2ENR
```

RCC AHB2 peripheral clock enable register

5.8.2.5 AHB2LPENR

```
__vo uint32_t RCC_RegDef_t::AHB2LPENR
```

RCC AHB2 peripheral clock enable in low power mode register

5.8.2.6 AHB2RSTR

```
__vo uint32_t RCC_RegDef_t::AHB2RSTR
```

RCC AHB2 peripheral reset register

5.8.2.7 AHB3ENR

```
__vo uint32_t RCC_RegDef_t::AHB3ENR
```

RCC AHB3 peripheral clock enable register

5.8.2.8 AHB3LPENR

```
__vo uint32_t RCC_RegDef_t::AHB3LPENR
```

RCC AHB3 peripheral clock enable in low power mode register

5.8.2.9 AHB3RSTR

```
__vo uint32_t RCC_RegDef_t::AHB3RSTR
```

RCC AHB3 peripheral reset register

5.8.2.10 APB1ENR

```
__vo uint32_t RCC_RegDef_t::APB1ENR
```

RCC APB1 peripheral clock enable register

5.8.2.11 APB1LPENR

```
__vo uint32_t RCC_RegDef_t::APB1LPENR
```

RCC APB1 peripheral clock enable in low power mode register

5.8.2.12 APB1RSTR

```
__vo uint32_t RCC_RegDef_t::APB1RSTR
```

RCC APB1 peripheral reset register

5.8.2.13 APB2ENR

```
__vo uint32_t RCC_RegDef_t::APB2ENR
```

RCC APB2 peripheral clock enable register

5.8.2.14 APB2LPENR

```
__vo uint32_t RCC_RegDef_t::APB2LPENR
```

RCC APB2 peripheral clock enable in low power mode register

5.8.2.15 APB2RSTR

```
__vo uint32_t RCC_RegDef_t::APB2RSTR
```

RCC APB2 peripheral reset register

5.8.2.16 BDCR

```
__vo uint32_t RCC_RegDef_t::BDCR
```

RCC backup domain control register

5.8.2.17 CFGR

```
__vo uint32_t RCC_RegDef_t::CFGR
```

RCC configuration register

5.8.2.18 CIR

```
__vo uint32_t RCC_RegDef_t::CIR
```

RCC clock configuration register

5.8.2.19 CKGATENR

```
__vo uint32_t RCC_RegDef_t::CKGATENR
```

RCC clocks gated enable register

5.8.2.20 CR

```
__vo uint32_t RCC_RegDef_t::CR
```

RCC control register

5.8.2.21 CSR

```
__vo uint32_t RCC_RegDef_t::CSR
```

RCC control/status register

5.8.2.22 DCKCFGR

```
__vo uint32_t RCC_RegDef_t::DCKCFGR
```

RCC dedicated clock configuration register

5.8.2.23 DCKCFGR2

```
__vo uint32_t RCC_RegDef_t::DCKCFGR2
```

RCC dedicated clocks configuration register 2

5.8.2.24 PLLCFGR

```
__vo uint32_t RCC_RegDef_t::PLLCFGR
```

RCC PLL configuration register

5.8.2.25 PLLI2SCFGR

```
__vo uint32_t RCC_RegDef_t::PLLI2SCFGR
```

RCC PLLI2S configuration register

5.8.2.26 PLLSAICFGR

```
__vo uint32_t RCC_RegDef_t::PLLSAICFGR
```

RCC PLLSAI configuration register

5.8.2.27 SSCGR

```
__vo uint32_t RCC_RegDef_t::SSCGR
```

RCC spread spectrum clock generation register

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

- drivers/Inc/stm32f407xx.h

5.9 RTC_date_t Struct Reference

Data structure for holding date information.

```
#include <ds1307.h>
```

Public Attributes

- uint8_t date
- uint8_t month
- uint8_t year
- uint8_t day

5.9.1 Detailed Description

Data structure for holding date information.

5.9.2 Member Data Documentation

5.9.2.1 date

```
uint8_t RTC_date_t::date
```

Day of the month (1-31).

5.9.2.2 day

```
uint8_t RTC_date_t::day
```

Day of the week (1-7, where 1 is Sunday).

5.9.2.3 month

```
uint8_t RTC_date_t::month
```

Month (1-12).

5.9.2.4 year

```
uint8_t RTC_date_t::year
```

Year (0-99).

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

- bsp/ds1307.h

5.10 RTC_time_t Struct Reference

Data structure for holding time information.

```
#include <ds1307.h>
```

Public Attributes

- uint8_t seconds
- uint8_t minutes
- uint8_t hours
- uint8_t time_format

5.10.1 Detailed Description

Data structure for holding time information.

5.10.2 Member Data Documentation

5.10.2.1 hours

```
uint8_t RTC_time_t::hours
```

Hours (0-23).

5.10.2.2 minutes

```
uint8_t RTC_time_t::minutes
```

Minutes (0-59).

5.10.2.3 seconds

```
uint8_t RTC_time_t::seconds
```

Seconds (0-59).

5.10.2.4 time_format

```
uint8_t RTC_time_t::time_format
```

Time format (12HRS_AM, 12HRS_PM, 24HRS).

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

- bsp/ds1307.h

5.11 SPI_Config_t Struct Reference

Configuration structure for SPI peripheral.

```
#include <stm32f407xx_spi_driver.h>
```

Public Attributes

- uint8_t SPI_DeviceMode
- uint8_t SPI_BusConfig
- uint8_t SPI_SclkSpeed
- uint8_t SPI_DFF
- uint8_t SPI_CPOL
- uint8_t SPI_CPHA
- uint8_t SPI_SSM

5.11.1 Detailed Description

Configuration structure for SPI peripheral.

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

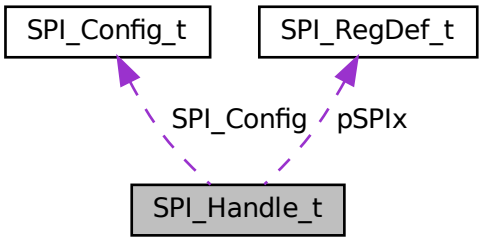
- drivers/Inc/stm32f407xx_spi_driver.h

5.12 SPI_Handle_t Struct Reference

Handle structure for SPI peripheral.

`#include <stm32f407xx_spi_driver.h>`

Collaboration diagram for SPI_Handle_t:



Public Attributes

- SPI_RegDef_t * pSPIx
- SPI_Config_t SPI_Config
- uint8_t * pTxBuffer
- uint8_t * pRxBuffer
- uint32_t TxLen
- uint32_t RxLen
- uint8_t TxState
- uint8_t RxState

5.12.1 Detailed Description

Handle structure for SPI peripheral.

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

- drivers/Inc/stm32f407xx_spi_driver.h

5.13 SPI_RegDef_t Struct Reference

SPI peripheral register definition structure.

`#include <stm32f407xx.h>`

Public Attributes

- __vo uint32_t **CR1**
- __vo uint32_t **CR2**
- __vo uint32_t **SR**
- __vo uint32_t **DR**
- __vo uint32_t **CRCPR**
- __vo uint32_t **RXCRCR**
- __vo uint32_t **TXCRCR**
- __vo uint32_t **I2SCFGR**
- __vo uint32_t **I2SPR**

5.13.1 Detailed Description

SPI peripheral register definition structure.

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

- drivers/Inc/stm32f407xx.h

5.14 SYSCFG_RegDef_t Struct Reference

SYSCFG peripheral register definition structure.

`#include <stm32f407xx.h>`

Public Attributes

- __vo uint32_t MEMRMP
- __vo uint32_t PMC
- __vo uint32_t EXTICR [4]
- uint32_t RESERVED1 [2]
- __vo uint32_t CMPCR
- uint32_t RESERVED2 [2]
- __vo uint32_t CFGR

5.14.1 Detailed Description

SYSCFG peripheral register definition structure.

5.14.2 Member Data Documentation

5.14.2.1 CFGR

```
__vo uint32_t SYSCFG_RegDef_t::CFGR
```

SYSCFG configuration register (Offset: 0x2C)

5.14.2.2 CMPCR

```
__vo uint32_t SYSCFG_RegDef_t::CMPCR
```

SYSCFG Compensation cell control register (Offset: 0x20)

5.14.2.3 EXTICR

```
__vo uint32_t SYSCFG_RegDef_t::EXTICR[4]
```

SYSCFG external interrupt configuration registers (Offset: 0x08 - 0x14)

5.14.2.4 MEMRMP

```
__vo uint32_t SYSCFG_RegDef_t::MEMRMP
```

SYSCFG memory remap register (Offset: 0x00)

5.14.2.5 PMC

```
__vo uint32_t SYSCFG_RegDef_t::PMC
```

SYSCFG peripheral mode configuration register (Offset: 0x04)

5.14.2.6 RESERVED1

```
uint32_t SYSCFG_RegDef_t::RESERVED1[2]
```

Reserved bits (Offset: 0x18 - 0x1F)

5.14.2.7 RESERVED2

```
uint32_t SYSCFG_RegDef_t::RESERVED2[2]
```

Reserved bits (Offset: 0x24 - 0x2B)

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

- drivers/Inc/stm32f407xx.h

5.15 USART_Config_t Struct Reference

Configuration structure for USART (Universal Synchronous Asynchronous Receiver Transmitter) peripheral.

`#include <stm32f407xx_usart_driver.h>`

Public Attributes

- uint8_t USART_Mode
- uint8_t USART_Baud
- uint8_t USART_NoOfStopBits
- uint8_t USART_WordLength
- uint8_t USART_ParityControl
- uint8_t USART_HWFlowControl

5.15.1 Detailed Description

Configuration structure for USART (Universal Synchronous Asynchronous Receiver Transmitter) peripheral.

5.15.2 Member Data Documentation

5.15.2.1 USART_Baud

`uint8_t USART_Config_t::USART_Baud`

Specifies the baud rate for serial communication.

5.15.2.2 USART_HWFlowControl

`uint8_t USART_Config_t::USART_HWFlowControl`

Specifies the hardware flow control mode, if applicable.

5.15.2.3 USART_Mode

`uint8_t USART_Config_t::USART_Mode`

Specifies the USART operating mode (e.g., transmit, receive, or both).

5.15.2.4 USART_NoOfStopBits

`uint8_t USART_Config_t::USART_NoOfStopBits`

Specifies the number of stop bits to use in each frame.

5.15.2.5 USART_ParityControl

```
uint8_t USART_Config_t::USART_ParityControl
```

Specifies the parity control mode for error checking.

5.15.2.6 USART_WordLength

```
uint8_t USART_Config_t::USART_WordLength
```

Specifies the word length (number of data bits) for each frame.

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

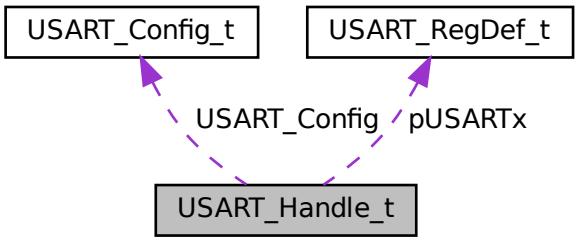
- drivers/Inc/stm32f407xx_usart_driver.h

5.16 USART_Handle_t Struct Reference

Handle structure for USART peripheral.

```
#include <stm32f407xx_usart_driver.h>
```

Collaboration diagram for USART_Handle_t:



Public Attributes

- USART_RegDef_t * pUSARTx
- USART_Config_t USART_Config
- uint8_t * pTxBuffer
- uint8_t * pRxBuffer
- uint32_t TxLen
- uint32_t RxLen
- uint8_t TxBusyState
- uint8_t RxBusyState

5.16.1 Detailed Description

Handle structure for USART peripheral.

5.16.2 Member Data Documentation

5.16.2.1 pRxBuffer

```
uint8_t* USART_Handle_t::pRxBuffer
```

Pointer to the receive buffer.

5.16.2.2 pTxBuffer

```
uint8_t* USART_Handle_t::pTxBuffer
```

Pointer to the transmit buffer.

5.16.2.3 pUSARTx

```
USART_RegDef_t* USART_Handle_t::pUSARTx
```

Pointer to the USART peripheral's base address.

5.16.2.4 RxBusyState

```
uint8_t USART_Handle_t::RxBusyState
```

Receiver state: BUSY (1) or IDLE (0).

5.16.2.5 RxLen

```
uint32_t USART_Handle_t::RxLen
```

Length of data to be received.

5.16.2.6 TxBusyState

```
uint8_t USART_Handle_t::TxBusyState
```

Transmitter state: BUSY (1) or IDLE (0).

5.16.2.7 TxLen

```
uint32_t USART_Handle_t::TxLen
```

Length of data to be transmitted.

5.16.2.8 USART_Config

```
USART_Config_t USART_Handle_t::USART_Config
```

USART configuration structure.

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

- drivers/Inc/stm32f407xx_usart_driver.h

5.17 USART_RegDef_t Struct Reference

USART peripheral register definition structure.

```
#include <stm32f407xx.h>
```

Public Attributes

- __vo uint32_t **SR**
- __vo uint32_t **DR**
- __vo uint32_t **BRR**
- __vo uint32_t **CR1**
- __vo uint32_t **CR2**
- __vo uint32_t **CR3**
- __vo uint32_t **GTPR**

5.17.1 Detailed Description

USART peripheral register definition structure.

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

- drivers/Inc/stm32f407xx.h

Chapter 6

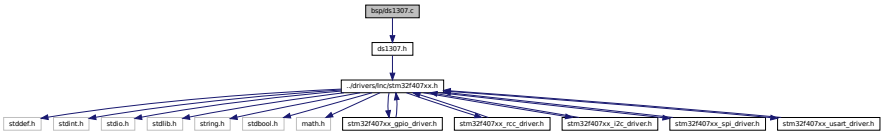
File Documentation

6.1 bsp/ds1307.c File Reference

This file contains the implementation of functions for interfacing with a DS1307 Real-Time Clock (RTC) module.

```
#include "ds1307.h"
```

Include dependency graph for ds1307.c:



Functions

- uint8_t ds1307_init ()

  Initialize the DS1307 module.
- void ds1307_set_current_time (RTC_time_t *rtc_time)

  Set the current time on the DS1307 RTC module.
- void ds1307_set_current_date (RTC_date_t *rtc_date)

  Set the current date on the DS1307 RTC module.
- void ds1307_get_current_time (RTC_time_t *rtc_time)

  Get the current time from the DS1307 RTC module.
- void ds1307_get_current_date (RTC_date_t *rtc_date)

  Get the current date from the DS1307 RTC module.

Variables

- I2C_Handle_t **g_ds1307I2cHandle**

6.1.1 Detailed Description

This file contains the implementation of functions for interfacing with a DS1307 Real-Time Clock (RTC) module.

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

This source file provides the implementation of functions for initializing, configuring, reading, and writing data to a DS1307 Real-Time Clock (RTC) module using the I2C communication protocol. It includes functions to set and retrieve time and date information, convert between binary and Binary-Coded Decimal (BCD) formats, and manage the DS1307's control registers. These functions facilitate the integration of DS1307 RTC functionality into embedded applications.

Copyright

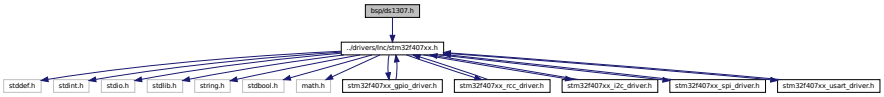
Copyright (c) 2023

6.2 bsp/ds1307.h File Reference

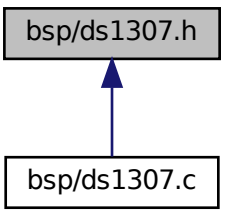
This file contains definitions and function prototypes for interfacing with a DS1307 Real-Time Clock (RTC) module.

```
#include "../drivers/Inc/stm32f407xx.h"
```

Include dependency graph for ds1307.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct RTC_date_t

  Data structure for holding date information.
- struct RTC_time_t

  Data structure for holding time information.

Macros

- #define **DS1307_I2C** I2C1
- #define **DS1307_I2C_GPIO_PORT** GPIOB
- #define **DS1307_I2C_SDA_PIN** GPIO_PIN_NO_7
- #define **DS1307_I2C_SCL_PIN** GPIO_PIN_NO_6
- #define **DS1307_I2C_SPEED** I2C_SC_SPEED_SM
- #define **DS1307_I2C_PUPD** GPIO_PIN_PU
- #define **DS1307_I2C_ADDRESS** 0x68
- #define **DS1307_ADDR_SEC** 0x00
- #define **DS1307_ADDR_MIN** 0x01
- #define **DS1307_ADDR_HRS** 0x02
- #define **DS1307_ADDR_DAY** 0x03
- #define **DS1307_ADDR_DATE** 0x04
- #define **DS1307_ADDR_MONTH** 0x05
- #define **DS1307_ADDR_YEAR** 0x06
- #define **TIME_FORMAT_12HRS_AM** 0
- #define **TIME_FORMAT_12HRS_PM** 1
- #define **TIME_FORMAT_24HRS** 2
- #define **SUNDAY** 1
- #define **MONDAY** 2
- #define **TUESDAY** 3
- #define **WEDNESDAY** 4
- #define **THURSDAY** 5
- #define **FRIDAY** 6
- #define **SATURDAY** 7

Functions

- uint8_t ds1307_init ()

  Initialize the DS1307 module.
- void ds1307_set_current_time (RTC_time_t *)

  Set the current time on the DS1307 RTC module.
- void ds1307_get_current_time (RTC_time_t *)

  Get the current time from the DS1307 RTC module.
- void ds1307_set_current_date (RTC_date_t *)

  Set the current date on the DS1307 RTC module.
- void ds1307_get_current_date (RTC_date_t *)

  Get the current date from the DS1307 RTC module.

6.2.1 Detailed Description

This file contains definitions and function prototypes for interfacing with a DS1307 Real-Time Clock (RTC) module.

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

Copyright

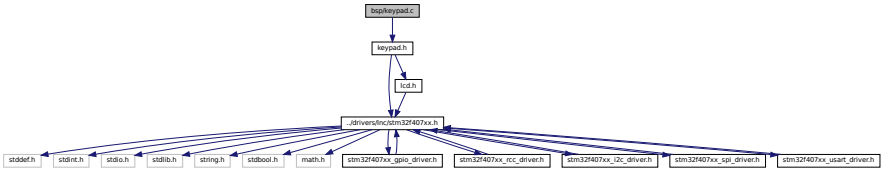
Copyright (c) 2023

6.3 bsp/keypad.c File Reference

Source file containing functions for interfacing with a keypad.

```
#include "keypad.h"
```

Include dependency graph for keypad.c:



Functions

- void Keypad_ScanAndPrint (uint8_t row_number, uint8_t row_pin, uint8_t column_pin, GPIO_RegDef_t *InputOutput_port)

  Scan and print the pressed key on the keypad.

Variables

- char * Keypad [4][4]

  Keypad key mappings.

6.3.1 Detailed Description

Source file containing functions for interfacing with a keypad.

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

Copyright

6.3.2 Variable Documentation

6.3.2.1 Keypad

```
char* Keypad[4][4]
```

Initial value:

```
= {
    {"Number: 1", "Number: 2", "Number: 3", "Character: A"},
    {"Number: 4", "Number: 5", "Number: 6", "Character: B"},
    {"Number: 7", "Number: 8", "Number: 9", "Character: C"},
    {"Character: *", "Number: 0", "Character: #", "Character: D"}
}
```

Keypad key mappings.

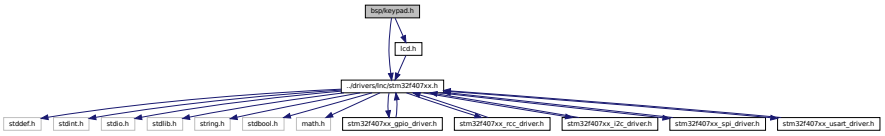
This 4x4 array defines the key mappings of the keypad.

6.4 bsp/keypad.h File Reference

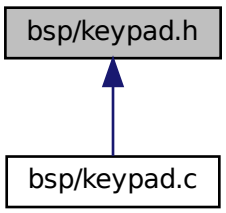
Header file containing definitions and function prototypes for interfacing with a keypad.

```
#include "../drivers/Inc/stm32f407xx.h"
#include "lcd.h"
```

Include dependency graph for keypad.h:



This graph shows which files directly or indirectly include this file:



Functions

- void Keypad_ScanAndPrint (uint8_t row_number, uint8_t row_pin, uint8_t column_pin, GPIO_RegDef_t *p↩ GPIOx)

  Scan and print the pressed key on the keypad.

6.4.1 Detailed Description

Header file containing definitions and function prototypes for interfacing with a keypad.

Version

0.1

Date

2023-10-07

Author

Mohamed Ali Haoufa

Copyright

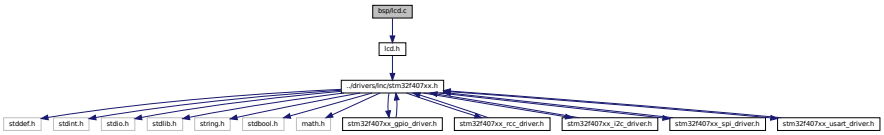
Copyright (c) 2023

6.5 bsp/lcd.c File Reference

This file contains the driver implementation for controlling an LCD (Liquid Crystal Display).

```
#include "lcd.h"
```

Include dependency graph for lcd.c:



Functions

- void lcd_send_command (uint8_t cmd)

  Send a command to the LCD.
- void lcd_print_char (uint8_t data)

  Print a character on the LCD.
- void lcd_print_string (char ∗message)

  Print a string on the LCD.
- void lcd_init (void)

  Initialize the LCD module.
- void lcd_set_cursor (uint8_t row, uint8_t column)

  Set the cursor position on the LCD.
- void lcd_display_clear (void)

  Clear the LCD display.
- void lcd_display_return_home (void)

  Return the cursor to the home position on the LCD.

6.5.1 Detailed Description

This file contains the driver implementation for controlling an LCD (Liquid Crystal Display).

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

This driver provides functions to initialize, control, and display text on an LCD using a 4-bit parallel data transmission method. It includes functions for sending commands, printing characters and strings, and clearing the display. The driver abstracts the hardware interactions, making it easy to interface with an LCD in embedded applications.

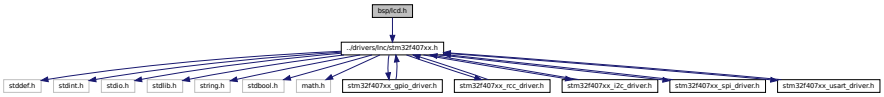
Copyright

6.6 bsp/lcd.h File Reference

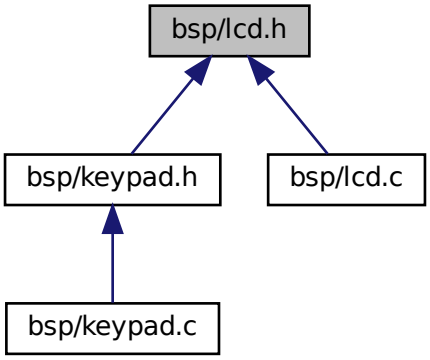
This file contains definitions and function prototypes for interfacing with an LCD (Liquid Crystal Display).

```
#include "../drivers/Inc/stm32f407xx.h"
```

Include dependency graph for lcd.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **LCD_GPIO_PORT** GPIOD
- #define **LCD_GPIO_RS** GPIO_PIN_NO_0
- #define **LCD_GPIO_RW** GPIO_PIN_NO_1
- #define **LCD_GPIO_EN** GPIO_PIN_NO_2
- #define **LCD_GPIO_D4** GPIO_PIN_NO_3
- #define **LCD_GPIO_D5** GPIO_PIN_NO_4
- #define **LCD_GPIO_D6** GPIO_PIN_NO_5
- #define **LCD_GPIO_D7** GPIO_PIN_NO_6
- #define **LCD_CMD_4DL_2N_5X8F** 0x28
- #define **LCD_CMD_DON_CON** 0x0E
- #define **LCD_CMD_DIS_CLEAR** 0x01
- #define **LCD_CMD_INCADD** 0x06
- #define **LCD_CMD_DIS_RETURN_HOME** 0x02

Functions

- void lcd_init (void)

  Initialize the LCD module.
- void lcd_send_command (uint8_t cmd)

  Send a command to the LCD.
- void lcd_print_char (uint8_t data)

  Print a character on the LCD.
- void lcd_print_string (char ∗message)

  Print a string on the LCD.
- void lcd_set_cursor (uint8_t row, uint8_t column)

  Set the cursor position on the LCD.
- void lcd_display_clear (void)

  Clear the LCD display.
- void lcd_display_return_home (void)

  Return the cursor to the home position on the LCD.

6.6.1 Detailed Description

This file contains definitions and function prototypes for interfacing with an LCD (Liquid Crystal Display).

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

This header file provides definitions for GPIO pins and LCD commands, as well as function prototypes for initializing, controlling, and displaying text on an LCD. It serves as an interface to abstract the low-level hardware interactions required for LCD operation, making it easier to integrate LCD functionality into embedded applications.

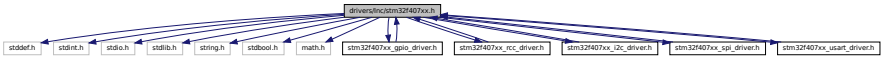
Copyright

6.7 drivers/Inc/stm32f407xx.h File Reference

Header file containing all the necessary information about the STM32F407xx MCU.

```
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <math.h>
#include "stm32f407xx_gpio_driver.h"
#include "stm32f407xx_rcc_driver.h"
#include "stm32f407xx_i2c_driver.h"
#include "stm32f407xx_spi_driver.h"
#include "stm32f407xx_usart_driver.h"
```

Include dependency graph for stm32f407xx.h:



This graph shows which files directly or indirectly include this file:

Classes

- struct GPIO_RegDef_t

  GPIO peripheral register definition structure.
- struct RCC_RegDef_t

  RCC peripheral register definition structure.
- struct EXTI_RegDef_t

  EXTI peripheral register definition structure.
- struct SYSCFG_RegDef_t

  SYSCFG peripheral register definition structure.
- struct SPI_RegDef_t

  SPI peripheral register definition structure.
- struct I2C_RegDef_t

  I2C peripheral register definition structure.
- struct USART_RegDef_t

  USART peripheral register definition structure.

Macros

- #define __vo volatile

  volatile_32bit_register Define for accessing a volatile 32-bit register
- #define __weak __attribute__((weak))

  Define for the weak attribute.
- #define **NVIC_ISER0** ((__vo uint32_t*)0xE000E100)
- #define **NVIC_ISER1** ((__vo uint32_t*)0xE000E104)
- #define **NVIC_ISER2** ((__vo uint32_t*)0xE000E108)
- #define **NVIC_ISER3** ((__vo uint32_t*)0xE000E10C)
- #define **NVIC_ISER4** ((__vo uint32_t*)0xE000E110)
- #define **NVIC_ISER5** ((__vo uint32_t*)0xE000E114)
- #define **NVIC_ISER6** ((__vo uint32_t*)0xE000E118)
- #define **NVIC_ISER7** ((__vo uint32_t*)0xE000E11C)
- #define **NVIC_ICER0** ((__vo uint32_t*)0xE000E180)
- #define **NVIC_ICER1** ((__vo uint32_t*)0xE000E184)
- #define **NVIC_ICER2** ((__vo uint32_t*)0xE000E188)
- #define **NVIC_ICER3** ((__vo uint32_t*)0xE000E18C)
- #define **NVIC_ICER4** ((__vo uint32_t*)0xE000E190)
- #define **NVIC_ICER5** ((__vo uint32_t*)0xE000E194)
- #define **NVIC_ICER6** ((__vo uint32_t*)0xE000E198)
- #define **NVIC_ICER7** ((__vo uint32_t*)0xE000E19C)
- #define **NVIC_PR_BASE_ADDR** ((__vo uint32_t*)0xE000E400)
- #define **NO_PR_BITS_IMPLEMENTED** 4
- #define FLASH_BASEADDR 0x08000000U
- #define SRAM1_BASEADDR (uint32_t)0x20000000
- #define **SRAM** SRAM1_BASE_ADDR
- #define ROM_BASEADDR 0x1FFF0000U
- #define SRAM2_BASEADDR 0x2001C000U
- #define PERIPH_BASEADDR 0x40000000U
- #define APB1PERIPH_BASEADDR PERIPH_BASEADDR
- #define APB2PERIPH_BASEADDR 0x40010000U
- #define AHB1PERIPH_BASEADDR 0x40020000U
- #define AHB2PERIPH_BASEADDR 0x50000000U
- #define GPIOA_BASEADDR (AHB1PERIPH_BASEADDR + 0x0000)
- #define GPIOB_BASEADDR (AHB1PERIPH_BASEADDR + 0x0400)
- #define GPIOC_BASEADDR (AHB1PERIPH_BASEADDR + 0x0800)
- #define GPIOD_BASEADDR (AHB1PERIPH_BASEADDR + 0x0C00)
- #define GPIOE_BASEADDR (AHB1PERIPH_BASEADDR + 0x1000)
- #define GPIOF_BASEADDR (AHB1PERIPH_BASEADDR + 0x1400)
- #define GPIOG_BASEADDR (AHB1PERIPH_BASEADDR + 0x1800)
- #define GPIOH_BASEADDR (AHB1PERIPH_BASEADDR + 0x1C00)
- #define GPIOI_BASEADDR (AHB1PERIPH_BASEADDR + 0x2000)
- #define RCC_BASEADDR (AHB1PERIPH_BASEADDR + 0x3800)
- #define DMA1_BASEADDR (AHB1PERIPH_BASEADDR + 0x6000)
- #define DMA2_BASEADDR (AHB1PERIPH_BASEADDR + 0x6400)
- #define CRC_BASEADDR (AHB1PERIPH_BASEADDR + 0x3000)
- #define FIR_BASEADDR (AHB1PERIPH_BASEADDR + 0x3C00)
- #define I2C1_BASEADDR (APB1PERIPH_BASEADDR+0x5400)
- #define I2C2_BASEADDR (APB1PERIPH_BASEADDR+0x5800)
- #define I2C3_BASEADDR (APB1PERIPH_BASEADDR+0x5C00)
- #define SPI2_BASEADDR (APB1PERIPH_BASEADDR+0x3800)
- #define SPI3_BASEADDR (APB1PERIPH_BASEADDR+0x3C00)
- #define USART2_BASEADDR (APB1PERIPH_BASEADDR+0x4400)

- #define USART3_BASEADDR (APB1PERIPH_BASEADDR+0x4800)
- #define UART4_BASEADDR (APB1PERIPH_BASEADDR+0x4C00)
- #define UART5_BASEADDR (APB1PERIPH_BASEADDR+0x5000)
- #define SPI1_BASEADDR (APB2PERIPH_BASEADDR+0x3000)
- #define SPI4_BASEADDR (APB2PERIPH_BASEADDR+0x3400)
- #define USART1_BASEADDR (APB2PERIPH_BASEADDR+0x1000)
- #define USART6_BASEADDR (APB2PERIPH_BASEADDR+0x1400)
- #define SYSCFG_BASEADDR (APB2PERIPH_BASEADDR+0x3800)
- #define EXTI_BASEADDR (APB2PERIPH_BASEADDR+0x3C00)
- #define **GPIOA** ((GPIO_RegDef_t*)GPIOA_BASEADDR)
- #define **GPIOB** ((GPIO_RegDef_t*)GPIOB_BASEADDR)
- #define **GPIOC** ((GPIO_RegDef_t*)GPIOC_BASEADDR)
- #define **GPIOD** ((GPIO_RegDef_t*)GPIOD_BASEADDR)
- #define **GPIOE** ((GPIO_RegDef_t*)GPIOE_BASEADDR)
- #define **GPIOF** ((GPIO_RegDef_t*)GPIOF_BASEADDR)
- #define **GPIOG** ((GPIO_RegDef_t*)GPIOG_BASEADDR)
- #define **GPIOH** ((GPIO_RegDef_t*)GPIOH_BASEADDR)
- #define **GPIOI** ((GPIO_RegDef_t*)GPIOI_BASEADDR)
- #define **RCC** ((RCC_RegDef_t*)RCC_BASEADDR)
- #define **EXTI** ((EXTI_RegDef_t*)EXTI_BASEADDR)
- #define **SYSCFG** ((SYSCFG_RegDef_t*)SYSCFG_BASEADDR)
- #define **SPI1** ((SPI_RegDef_t*)SPI1_BASEADDR)
- #define **SPI2** ((SPI_RegDef_t*)SPI2_BASEADDR)
- #define **SPI3** ((SPI_RegDef_t*)SPI3_BASEADDR)
- #define **SPI4** ((SPI_RegDef_t*)SPI4_BASEADDR)
- #define **I2C1** ((I2C_RegDef_t*)I2C1_BASEADDR)
- #define **I2C2** ((I2C_RegDef_t*)I2C2_BASEADDR)
- #define **I2C3** ((I2C_RegDef_t*)I2C3_BASEADDR)
- #define **USART1** ((USART_RegDef_t*)USART1_BASEADDR)
- #define **USART2** ((USART_RegDef_t*)USART2_BASEADDR)
- #define **USART3** ((USART_RegDef_t*)USART3_BASEADDR)
- #define **UART4** ((USART_RegDef_t*)UART4_BASEADDR)
- #define **UART5** ((USART_RegDef_t*)UART5_BASEADDR)
- #define **USART6** ((USART_RegDef_t*)USART6_BASEADDR)
- #define **GPIOA_PCLK_EN**() (RCC->AHB1ENR |= (1<<0))
- #define **GPIOB_PCLK_EN**() (RCC->AHB1ENR |= (1<<1))
- #define **GPIOC_PCLK_EN**() (RCC->AHB1ENR |= (1<<2))
- #define **GPIOD_PCLK_EN**() (RCC->AHB1ENR |= (1<<3))
- #define **GPIOE_PCLK_EN**() (RCC->AHB1ENR |= (1<<4))
- #define **GPIOF_PCLK_EN**() (RCC->AHB1ENR |= (1<<5))
- #define **GPIOG_PCLK_EN**() (RCC->AHB1ENR |= (1<<6))
- #define **GPIOH_PCLK_EN**() (RCC->AHB1ENR |= (1<<7))
- #define **GPIOI_PCLK_EN**() (RCC->AHB1ENR |= (1<<8))
- #define **I2C1_PCLK_EN**() (RCC->APB1ENR |= (1<<21))
- #define **I2C2_PCLK_EN**() (RCC->APB1ENR |= (1<<22))
- #define **I2C3_PCLK_EN**() (RCC->APB1ENR |= (1<<23))
- #define **SPI1_PCLK_EN**() (RCC->APB2ENR |= (1<<12))
- #define **SPI2_PCLK_EN**() (RCC->APB1ENR |= (1<<14))
- #define **SPI3_PCLK_EN**() (RCC->APB1ENR |= (1<<15))
- #define **SPI4_PCLK_EN**() (RCC->APB2ENR |= (1<<13))
- #define **USART1_PCLK_EN**() (RCC->APB2ENR |= (1<<4))
- #define **USART2_PCLK_EN**() (RCC->APB1ENR |= (1<<17))
- #define **USART3_PCLK_EN**() (RCC->APB1ENR |= (1<<18))
- #define **UART4_PCLK_EN**() (RCC->APB1ENR |= (1<<19))
- #define **UART5_PCLK_EN**() (RCC->APB1ENR |= (1<<20))

- #define **USART6_PCLK_EN**() (RCC->APB2ENR |= (1<<5))
- #define **SYSCFG_PCLK_EN**() (RCC->APB2ENR |= (1<<14))
- #define **GPIOA_PCLK_DI**() (RCC->AHB1ENR &= ~(uint32_t)(1<<0))
- #define **GPIOB_PCLK_DI**() (RCC->AHB1ENR &= ~(uint32_t)(1<<1))
- #define **GPIOC_PCLK_DI**() (RCC->AHB1ENR &= ~(uint32_t)(1<<2))
- #define **GPIOD_PCLK_DI**() (RCC->AHB1ENR &= ~(uint32_t)(1<<3))
- #define **GPIOE_PCLK_DI**() (RCC->AHB1ENR &= ~(uint32_t)(1<<4))
- #define **GPIOF_PCLK_DI**() (RCC->AHB1ENR &= ~(uint32_t)(1<<5))
- #define **GPIOG_PCLK_DI**() (RCC->AHB1ENR &= ~(uint32_t)(1<<6))
- #define **GPIOH_PCLK_DI**() (RCC->AHB1ENR &= ~(uint32_t)(1<<7))
- #define **GPIOI_PCLK_DI**() (RCC->AHB1ENR &= ~(uint32_t)(1<<8))
- #define **I2C1_PCLK_DI**() (RCC->APB1ENR &= ~(uint32_t)(1<<21))
- #define **I2C2_PCLK_DI**() (RCC->APB1ENR &= ~(uint32_t)(1<<22))
- #define **I2C3_PCLK_DI**() (RCC->APB1ENR &= ~(uint32_t)(1<<23))
- #define **SPI1_PCLK_DI**() (RCC->APB2ENR &= ~(uint32_t)(1<<12))
- #define **SPI2_PCLK_DI**() (RCC->APB1ENR &= ~(uint32_t)(1<<14))
- #define **SPI3_PCLK_DI**() (RCC->APB1ENR &= ~(uint32_t)(1<<15))
- #define **SPI4_PCLK_DI**() (RCC->APB2ENR &= ~(uint32_t)(1<<13))
- #define **USART1_PCLK_DI**() (RCC->APB2ENR &= ~(uint32_t)(1<<4))
- #define **USART2_PCLK_DI**() (RCC->APB1ENR &= ~(uint32_t)(1<<17))
- #define **USART3_PCLK_DI**() (RCC->APB1ENR &= ~(uint32_t)(1<<18))
- #define **UART4_PCLK_DI**() (RCC->APB1ENR &= ~(uint32_t)(1<<19))
- #define **UART5_PCLK_DI**() (RCC->APB1ENR &= ~(uint32_t)(1<<20))
- #define **USART6_PCLK_DI**() (RCC->APB2ENR &= ~(uint32_t)(1<<5))
- #define **SYSCFG_PCLK_DI**() (RCC->APB2ENR &= ~(uint32_t)(1<<14))
- #define **GPIOA_REG_RESET**() do{ (RCC->AHB1RSTR |= (1<<0)); (RCC->AHB1RSTR &= ~(1<<0));}while(0)
- #define **GPIOB_REG_RESET**() do{ (RCC->AHB1RSTR |= (1<<1)); (RCC->AHB1RSTR &= ~(1<<1));}while(0)
- #define **GPIOC_REG_RESET**() do{ (RCC->AHB1RSTR |= (1<<2)); (RCC->AHB1RSTR &= ~(1<<2));}while(0)
- #define **GPIOD_REG_RESET**() do{ (RCC->AHB1RSTR |= (1<<3)); (RCC->AHB1RSTR &= ~(1<<3));}while(0)
- #define **GPIOE_REG_RESET**() do{ (RCC->AHB1RSTR |= (1<<4)); (RCC->AHB1RSTR &= ~(1<<4));}while(0)
- #define **GPIOF_REG_RESET**() do{ (RCC->AHB1RSTR |= (1<<5)); (RCC->AHB1RSTR &= ~(1<<5));}while(0)
- #define **GPIOG_REG_RESET**() do{ (RCC->AHB1RSTR |= (1<<6)); (RCC->AHB1RSTR &= ~(1<<6));}while(0)
- #define **GPIOH_REG_RESET**() do{ (RCC->AHB1RSTR |= (1<<7)); (RCC->AHB1RSTR &= ~(1<<7));}while(0)
- #define **GPIOI_REG_RESET**() do{ (RCC->AHB1RSTR |= (1<<8)); (RCC->AHB1RSTR &= ~(1<<8));}while(0)
- #define **I2C1_REG_RESET**() do{ (RCC->APB1RSTR |= (1<<21)); (RCC->APB1RSTR &= ~(1<<21));}while(0)
- #define **I2C2_REG_RESET**() do{ (RCC->APB1RSTR |= (1<<22)); (RCC->APB1RSTR &= ~(1<<22));}while(0)
- #define **I2C3_REG_RESET**() do{ (RCC->APB1RSTR |= (1<<23)); (RCC->APB1RSTR &= ~(1<<23));}while(0)
- #define **SPI1_REG_RESET**() do{ (RCC->APB2RSTR |= (1<<12)); (RCC->APB2RSTR &= ~(1<<12));}while(0)
- #define **SPI2_REG_RESET**() do{ (RCC->APB1RSTR |= (1<<14)); (RCC->APB1RSTR &= ~(1<<14));}while(0)
- #define **SPI3_REG_RESET**() do{ (RCC->APB1RSTR |= (1<<15)); (RCC->APB1RSTR &= ~(1<<15));}while(0)

- #define **SPI4_REG_RESET**() do{ (RCC->APB2RSTR |= (1<<13)); (RCC->APB2RSTR &= ~(1<<13));}while(0)
- #define **USART1_REG_RESET**() do{ (RCC->APB2RSTR |= (1<<4)); (RCC->APB2RSTR &= ~(1<<4));}while(0)
- #define **USART2_REG_RESET**() do{ (RCC->APB1RSTR |= (1<<17)); (RCC->APB1RSTR &= ~(1<<17));}while(0)
- #define **USART3_REG_RESET**() do{ (RCC->APB1RSTR |= (1<<18)); (RCC->APB1RSTR &= ~(1<<18));}while(0)
- #define **UART4_REG_RESET**() do{ (RCC->APB1RSTR |= (1<<19)); (RCC->APB1RSTR &= ~(1<<19));}while(0)
- #define **UART5_REG_RESET**() do{ (RCC->APB1RSTR |= (1<<20)); (RCC->APB1RSTR &= ~(1<<20));}while(0)
- #define **USART6_REG_RESET**() do{ (RCC->APB2RSTR |= (1<<5)); (RCC->APB2RSTR &= ~(1<<5));}while(0)
- #define GPIO_BASEADDR_TO_CODE(x)

  Macro to convert GPIO base address to port code.
- #define **IRQ_NO_EXTI0** 6
- #define **IRQ_NO_EXTI1** 7
- #define **IRQ_NO_EXTI2** 8
- #define **IRQ_NO_EXTI3** 9
- #define **IRQ_NO_EXTI4** 10
- #define **IRQ_NO_EXTI9_5** 23
- #define **IRQ_NO_EXTI15_10** 40
- #define **IRQ_NO_SPI1** 35
- #define **IRQ_NO_SPI2** 36
- #define **IRQ_NO_SPI3** 51
- #define **IRQ_NO_SPI4** 84
- #define **IRQ_NO_I2C1_EV** 31
- #define **IRQ_NO_I2C1_ER** 32
- #define **IRQ_NO_I2C2_EV** 33
- #define **IRQ_NO_I2C2_ER** 34
- #define **IRQ_NO_I2C3_EV** 79
- #define **IRQ_NO_I2C3_ER** 80
- #define **IRQ_NO_USART1** 37
- #define **IRQ_NO_USART2** 38
- #define **IRQ_NO_USART3** 39
- #define **IRQ_NO_UART4** 52
- #define **IRQ_NO_UART5** 53
- #define **IRQ_NO_USART6** 71
- #define **NVIC_IRQ_PRI0** 0
- #define **NVIC_IRQ_PRI1** 1
- #define **NVIC_IRQ_PRI2** 2
- #define **NVIC_IRQ_PRI3** 3
- #define **NVIC_IRQ_PRI4** 4
- #define **NVIC_IRQ_PRI5** 5
- #define **NVIC_IRQ_PRI6** 6
- #define **NVIC_IRQ_PRI7** 7
- #define **NVIC_IRQ_PRI8** 8
- #define **NVIC_IRQ_PRI9** 9
- #define **NVIC_IRQ_PRI10** 10
- #define **NVIC_IRQ_PRI11** 11
- #define **NVIC_IRQ_PRI12** 12
- #define **NVIC_IRQ_PRI13** 13
- #define **NVIC_IRQ_PRI14** 14
- #define **NVIC_IRQ_PRI15** 15

- #define SPI_CR1_CPHA 0
- #define SPI_CR1_CPOL 1
- #define SPI_CR1_MSTR 2
- #define SPI_CR1_BR 3
- #define SPI_CR1_SPE 6
- #define SPI_CR1_LSBFIRST 7
- #define SPI_CR1_SSI 8
- #define SPI_CR1_SSM 9
- #define SPI_CR1_RXONLY 10
- #define SPI_CR1_DFF 11
- #define SPI_CR1_CRCNEXT 12
- #define SPI_CR1_CRCEN 13
- #define SPI_CR1_BIDIOE 14
- #define SPI_CR1_BIDIMODE 15
- #define SPI_CR2_RXDMAEN 0
- #define SPI_CR2_TXDMAEN 1
- #define SPI_CR2_SSOE 2
- #define SPI_CR2_FRF 4
- #define SPI_CR2_ERRIE 5
- #define SPI_CR2_RXNEIE 6
- #define SPI_CR2_TXEIE 7
- #define SPI_SR_RXNE 0
- #define SPI_SR_TXE 1
- #define SPI_SR_CHSIDE 2
- #define SPI_SR_UDR 3
- #define SPI_SR_CRCERR 4
- #define SPI_SR_MODF 5
- #define SPI_SR_OVR 6
- #define SPI_SR_BSY 7
- #define SPI_SR_FRE 8
- #define I2C_CR1_PE 0
- #define I2C_CR1_SMBUS 1
- #define I2C_CR1_SMBTYPE 3
- #define I2C_CR1_ENARP 4
- #define I2C_CR1_ENPEC 5
- #define I2C_CR1_ENGC 6
- #define I2C_CR1_NOSTRETCH 7
- #define I2C_CR1_START 8
- #define I2C_CR1_STOP 9
- #define I2C_CR1_ACK 10
- #define I2C_CR1_POS 11
- #define I2C_CR1_PEC 12
- #define I2C_CR1_ALERT 13
- #define I2C_CR1_SWRST 15
- #define I2C_OAR1_ADD0 0
- #define I2C_OAR1_ADD 1
- #define I2C_OAR1_ADDMODE 15
- #define I2C_OAR2_ENDUAL 0
- #define I2C_OAR2_ADD2 1
- #define I2C_CR2_FREQ 0
- #define I2C_CR2_ITERREN 8
- #define I2C_CR2_ITEVTEN 9
- #define I2C_CR2_ITBUFEN 10
- #define I2C_CR2_DMAEN 11
- #define I2C_CR2_LAST 12

- #define I2C_SR1_SB 0
- #define I2C_SR1_ADDR 1
- #define I2C_SR1_BTF 2
- #define I2C_SR1_ADD10 3
- #define I2C_SR1_STOPF 4
- #define I2C_SR1_RxNE 6
- #define I2C_SR1_TxE 7
- #define I2C_SR1_BERR 8
- #define I2C_SR1_ARLO 9
- #define I2C_SR1_AF 10
- #define I2C_SR1_OVR 11
- #define I2C_SR1_PECERR 12
- #define I2C_SR1_TIMEOUT 14
- #define I2C_SR1_SMBALERT 15
- #define I2C_SR2_MSL 0
- #define I2C_SR2_BUSY 1
- #define I2C_SR2_TRA 2
- #define I2C_SR2_GENCALL 4
- #define I2C_SR2_SMBDEFAULT 5
- #define I2C_SR2_SMBHOST 6
- #define I2C_SR2_DUALF 7
- #define I2C_SR2_PEC 8
- #define I2C_CCR_CCR 0
- #define I2C_CCR_DUTY 14
- #define I2C_CCR_FS 15
- #define I2C_TRISE_TRISE 0
- #define I2C_FLTR_DNF 0
- #define I2C_FLTR_ANOFF 4
- #define USART_SR_PE 0
- #define USART_SR_FE 1
- #define USART_SR_NF 2
- #define USART_SR_ORE 3
- #define USART_SR_IDLE 4
- #define USART_SR_RXNE 5
- #define USART_SR_TC 6
- #define USART_SR_TXE 7
- #define USART_SR_LBD 8
- #define USART_SR_CTS 9
- #define USART_DR_DR 0
- #define USART_BRR_DIV_FRACTION 0
- #define USART_BRR_DIV_MANTISSA 4
- #define USART_CR1_SBK 0
- #define USART_CR1_RWU 1
- #define USART_CR1_RE 2
- #define USART_CR1_TE 3
- #define USART_CR1_IDLEIE 4
- #define USART_CR1_RXNEIE 5
- #define USART_CR1_TCIE 6
- #define USART_CR1_TXEIE 7
- #define USART_CR1_PEIE 8
- #define USART_CR1_PS 9
- #define USART_CR1_PCE 10
- #define USART_CR1_WAKE 11
- #define USART_CR1_M 12
- #define USART_CR1_UE 13

- #define USART_CR1_OVER8 15
- #define USART_CR2_ADD 4
- #define USART_CR2_LBDL 5
- #define USART_CR2_LBDIE 6
- #define USART_CR2_LBCL 8
- #define USART_CR2_CPHA 9
- #define USART_CR2_CPOL 10
- #define USART_CR2_CLKEN 11
- #define USART_CR2_STOP 12
- #define USART_CR2_LINEN 14
- #define USART_CR3_EIE 0
- #define USART_CR3_IREN 1
- #define USART_CR3_IRLP 2
- #define USART_CR3_HDSEL 3
- #define USART_CR3_NACK 4
- #define USART_CR3_SCEN 5
- #define USART_CR3_DMAR 6
- #define USART_CR3_DMAT 7
- #define USART_CR3_RTSE 8
- #define USART_CR3_CTSE 9
- #define USART_CR3_CTSIE 10
- #define USART_CR3_ONEBIT 11
- #define USART_GTPR_PSC 0
- #define USART_GTPR_GT 8
- #define RCC_CR_HSION 0
- #define RCC_CR_HSIRDY 1
- #define RCC_CR_HSITRIM 3
- #define RCC_CR_HSICAL 8
- #define RCC_CR_HSEON 16
- #define RCC_CR_HSERDY 17
- #define RCC_CR_HSEBYP 18
- #define RCC_CR_CSSON 19
- #define RCC_CR_PLLON 24
- #define RCC_CR_PLLRDY 25
- #define RCC_CR_PLLI2SON 26
- #define RCC_CR_PLLI2SRDY 27
- #define RCC_CR_PLLSAION 28
- #define RCC_CR_PLLSAIRDY 29
- #define RCC_PLLCFGR_PLLM 0
- #define RCC_PLLCFGR_PLLN 6
- #define RCC_PLLCFGR_PLLP 16
- #define RCC_PLLCFGR_PLLSRC 22
- #define RCC_PLLCFGR_PLLQ 24
- #define RCC_CFGR_SW 0
- #define RCC_CFGR_SWS 2
- #define RCC_CFGR_HPRE 4
- #define RCC_CFGR_PPRE1 10
- #define RCC_CFGR_PPRE2 13
- #define RCC_CFGR_RTCPRE 16
- #define RCC_CFGR_MCO1 21
- #define RCC_CFGR_I2SSRC 23
- #define RCC_CFGR_MCO1PRE 24
- #define RCC_CFGR_MCO2PRE 27
- #define RCC_CFGR_MCO2 30
- #define RCC_CIR_LSIRDYF 0

- #define RCC_CIR_LSERDYF 1
- #define RCC_CIR_HSIRDYF 2
- #define RCC_CIR_HSERDYF 3
- #define RCC_CIR_PLLRDYF 4
- #define RCC_CIR_PLLI2SRDYF 5
- #define RCC_CIR_PLLSAIRDYF 6
- #define RCC_CIR_CSSF 7
- #define RCC_CIR_LSIRDYIE 8
- #define RCC_CIR_LSERDYIE 9
- #define RCC_CIR_HSIRDYIE 10
- #define RCC_CIR_HSERDYIE 11
- #define RCC_CIR_PLLRDYIE 12
- #define RCC_CIR_PLLI2SRDYIE 13
- #define RCC_CIR_PLLSAIRDYIE 14
- #define RCC_CIR_LSIRDYC 16
- #define RCC_CIR_LSERDYC 17
- #define RCC_CIR_HSIRDYC 18
- #define RCC_CIR_HSERDYC 19
- #define RCC_CIR_PLLRDYC 20
- #define RCC_CIR_PLLI2SRDYC 21
- #define RCC_CIR_PLLSAIRDYC 22
- #define RCC_AHB1RSTR_GPIOA 0
- #define RCC_AHB1RSTR_GPIOB 1
- #define RCC_AHB1RSTR_GPIOC 2
- #define RCC_AHB1RSTR_GPIOD 3
- #define RCC_AHB1RSTR_GPIOE 4
- #define RCC_AHB1RSTR_GPIOF 5
- #define RCC_AHB1RSTR_GPIOG 6
- #define RCC_AHB1RSTR_GPIOH 7
- #define RCC_AHB1RSTR_GPIOI 8
- #define RCC_AHB1RSTR_CRC 12
- #define RCC_AHB1RSTR_DMA1 21
- #define RCC_AHB1RSTR_DMA2 22
- #define RCC_AHB1RSTR_ETHMAC 25
- #define RCC_AHB1RSTR_OTGHS 29
- #define RCC_AHB1RSTR_OTGHSULPI 30
- #define RCC_AHB2RSTR_DCMI 0
- #define RCC_AHB2RSTR_CRYP 4
- #define RCC_AHB2RSTR_HASH 5
- #define RCC_AHB2RSTR_RNG 6
- #define RCC_AHB2RSTR_OTGFS 7
- #define RCC_AHB3RSTR_FSMC 0
- #define RCC_APB1RSTR_TIM2 0
- #define RCC_APB1RSTR_TIM3 1
- #define RCC_APB1RSTR_TIM4 2
- #define RCC_APB1RSTR_TIM5 3
- #define RCC_APB1RSTR_TIM6 4
- #define RCC_APB1RSTR_TIM7 5
- #define RCC_APB1RSTR_TIM12 6
- #define RCC_APB1RSTR_TIM13 7
- #define RCC_APB1RSTR_TIM14 8
- #define RCC_APB1RSTR_WWDG 11
- #define RCC_APB1RSTR_SPI2 14
- #define RCC_APB1RSTR_SPI3 15
- #define RCC_APB1RSTR_USART2 17

- #define RCC_APB1RSTR_USART3 18
- #define RCC_APB1RSTR_UART4 19
- #define RCC_APB1RSTR_UART5 20
- #define RCC_APB1RSTR_I2C1 21
- #define RCC_APB1RSTR_I2C2 22
- #define RCC_APB1RSTR_I2C3 23
- #define RCC_APB1RSTR_CAN1 25
- #define RCC_APB1RSTR_CAN2 26
- #define RCC_APB1RSTR_PWR 28
- #define RCC_APB1RSTR_DAC 29
- #define RCC_APB1RSTR_UART7 30
- #define RCC_APB1RSTR_UART8 31
- #define RCC_APB2RSTR_TIM1 0
- #define RCC_APB2RSTR_TIM8 1
- #define RCC_APB2RSTR_USART1 4
- #define RCC_APB2RSTR_USART6 5
- #define RCC_APB2RSTR_ADC 8
- #define RCC_APB2RSTR_SDIO 11
- #define RCC_APB2RSTR_SPI1 12
- #define RCC_APB2RSTR_SYSCFG 14
- #define RCC_APB2RSTR_TIM9 16
- #define RCC_APB2RSTR_TIM10 17
- #define RCC_APB2RSTR_TIM11 18
- #define RCC_AHB1LPENR_GPIOALPEN 0
- #define RCC_AHB1LPENR_GPIOBLPEN 1
- #define RCC_AHB1LPENR_GPIOCLPEN 2
- #define RCC_AHB1LPENR_GPIODLPEN 3
- #define RCC_AHB1LPENR_GPIOELPEN 4
- #define RCC_AHB1LPENR_GPIOFLPEN 5
- #define RCC_AHB1LPENR_GPIOGLPEN 6
- #define RCC_AHB1LPENR_GPIOHLPEN 7
- #define RCC_AHB1LPENR_GPIOILPEN 8
- #define RCC_AHB1LPENR_CRCEN 12
- #define RCC_AHB1LPENR_DMA1LPEN 21
- #define RCC_AHB1LPENR_DMA2LPEN 22
- #define RCC_AHB1LPENR_ETHMACLPEN 25
- #define RCC_AHB1LPENR_ETHMACTXLPEN 26
- #define RCC_AHB1LPENR_ETHMACRXLPEN 27
- #define RCC_AHB1LPENR_ETHMACPTPLPEN 28
- #define RCC_AHB1LPENR_OTGHSLPEN 29
- #define RCC_AHB1LPENR_OTGHSHULPI 30
- #define RCC_AHB2LPENR_DCMILPEN 0
- #define RCC_AHB2LPENR_CRYPLPEN 4
- #define RCC_AHB2LPENR_HASHLPEN 5
- #define RCC_AHB2LPENR_RNGLPEN 6
- #define RCC_AHB2LPENR_OTGFSLPEN 7
- #define RCC_AHB3LPENR_FSMCLPEN 0
- #define RCC_APB1LPENR_TIM2LPEN 0
- #define RCC_APB1LPENR_TIM3LPEN 1
- #define RCC_APB1LPENR_TIM4LPEN 2
- #define RCC_APB1LPENR_TIM5LPEN 3
- #define RCC_APB1LPENR_TIM6LPEN 4
- #define RCC_APB1LPENR_TIM7LPEN 5
- #define RCC_APB1LPENR_TIM12LPEN 6
- #define RCC_APB1LPENR_TIM13LPEN 7

- #define RCC_APB1LPENR_TIM14LPEN 8
- #define RCC_APB1LPENR_WWDGLPEN 11
- #define RCC_APB1LPENR_SPI2LPEN 14
- #define RCC_APB1LPENR_SPI3LPEN 15
- #define RCC_APB1LPENR_USART2LPEN 17
- #define RCC_APB1LPENR_USART3LPEN 18
- #define RCC_APB1LPENR_UART4LPEN 19
- #define RCC_APB1LPENR_UART5LPEN 20
- #define RCC_APB1LPENR_I2C1LPEN 21
- #define RCC_APB1LPENR_I2C2LPEN 22
- #define RCC_APB1LPENR_I2C3LPEN 23
- #define RCC_APB1LPENR_CAN1LPEN 25
- #define RCC_APB1LPENR_CAN2LPEN 26
- #define RCC_APB1LPENR_PWRLPEN 28
- #define RCC_APB1LPENR_DACLPEN 29
- #define RCC_APB1LPENR_UART7LPEN 30
- #define RCC_APB1LPENR_UART8LPEN 31
- #define RCC_APB2LPENR_TIM1LPEN 0
- #define RCC_APB2LPENR_TIM8LPEN 1
- #define RCC_APB2LPENR_USART1LPEN 4
- #define RCC_APB2LPENR_USART6LPEN 5
- #define RCC_APB2LPENR_ADCLPEN 8
- #define RCC_APB2LPENR_SDIOLPEN 11
- #define RCC_APB2LPENR_SPI1LPEN 12
- #define RCC_APB2LPENR_SYSCFGLPEN 14
- #define RCC_APB2LPENR_TIM9LPEN 16
- #define RCC_APB2LPENR_TIM10LPEN 17
- #define RCC_APB2LPENR_TIM11LPEN 18
- #define RCC_BDCR_LSEON 0
- #define RCC_BDCR_LSERDY 1
- #define RCC_BDCR_LSEBYP 2
- #define RCC_BDCR_RTCSEL 8
- #define RCC_BDCR_RTCEN 15
- #define RCC_BDCR_BDRST 16
- #define RCC_CSR_LSION 0
- #define RCC_CSR_LSIRDY 1
- #define RCC_CSR_RMVF 24
- #define RCC_CSR_OBLRSTF 25
- #define RCC_CSR_PINRSTF 26
- #define RCC_CSR_PORRSTF 27
- #define RCC_CSR_SFTRSTF 28
- #define RCC_CSR_IWDGRSTF 29
- #define RCC_CSR_WWDGRSTF 30
- #define RCC_CSR_LPWRRSTF 31
- #define RCC_SSCGR_MODPER 0
- #define RCC_SSCGR_INCSTEP 13
- #define RCC_SSCGR_SPREADSEL 15
- #define RCC_SSCGR_SSCGEN 31
- #define RCC_PLLI2SCFGR_PLLI2SN 6
- #define RCC_PLLI2SCFGR_PLLI2SR 28
- #define ENABLE 1
- #define DISABLE 0
- #define SET ENABLE
- #define RESET DISABLE
- #define GPIO_PIN_SET SET
- #define GPIO_PIN_RESET RESET
- #define FLAG_SET SET
- #define FLAG_RESET RESET

6.7.1 Detailed Description

Header file containing all the necessary information about the STM32F407xx MCU.

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

This file contains the definitions and macros for the STM32F407xx microcontroller peripherals and memory maps.

Copyright

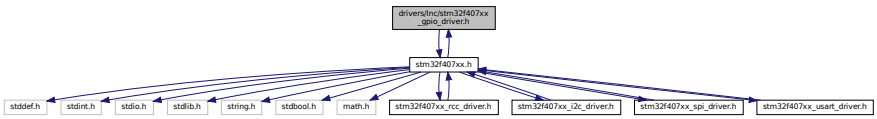
Copyright (c) 2023

6.8 drivers/Inc/stm32f407xx_gpio_driver.h File Reference

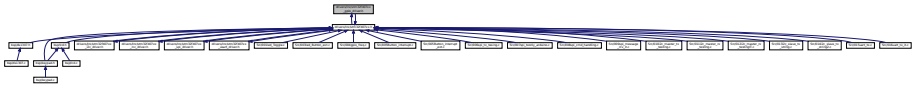
This file contains the GPIO driver API declarations for the STM32F407xx MCU.

```
#include "stm32f407xx.h"
```

Include dependency graph for stm32f407xx_gpio_driver.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct GPIO_PinConfig_t

  Configuration structure for GPIO pin.
- struct GPIO_Handle_t

  Handle structure for GPIO pin.

Macros

- #define **GPIO_PIN_NO_0** 0
- #define **GPIO_PIN_NO_1** 1
- #define **GPIO_PIN_NO_2** 2
- #define **GPIO_PIN_NO_3** 3
- #define **GPIO_PIN_NO_4** 4
- #define **GPIO_PIN_NO_5** 5
- #define **GPIO_PIN_NO_6** 6
- #define **GPIO_PIN_NO_7** 7
- #define **GPIO_PIN_NO_8** 8
- #define **GPIO_PIN_NO_9** 9
- #define **GPIO_PIN_NO_10** 10
- #define **GPIO_PIN_NO_11** 11
- #define **GPIO_PIN_NO_12** 12
- #define **GPIO_PIN_NO_13** 13
- #define **GPIO_PIN_NO_14** 14
- #define **GPIO_PIN_NO_15** 15
- #define GPIO_MODE_IN 0
- #define GPIO_MODE_OUT 1
- #define GPIO_MODE_ALTFN 2
- #define GPIO_MODE_ANALOG 3
- #define GPIO_MODE_IT_FT 4
- #define GPIO_MODE_IT_RT 5
- #define GPIO_MODE_IT_RFT 6
- #define GPIO_SPEED_LOW 0
- #define GPIO_SPEED_MEDIUM 1
- #define GPIO_SPEED_FAST 2
- #define GPIO_SPEED_HIGH 3
- #define GPIO_NO_PUPD 0
- #define GPIO_PIN_PU 1
- #define GPIO_PIN_PD 2
- #define GPIO_OP_TYPE_PP 0
- #define GPIO_OP_TYPE_OD 1

Functions

- void GPIO_PeripheralClockControl (GPIO_RegDef_t *pGPIOx, uint8_t EnorDi)

  Enables or disables the peripheral clock for the GPIO port.
- void GPIO_Init (GPIO_Handle_t *pGPIOHandle)

  Initializes the GPIO port pin according to the configuration.
- void GPIO_DeInit (GPIO_RegDef_t *pGPIOx)

  Deinitializes the GPIO port.
- uint8_t GPIO_ReadFromInputPin (GPIO_RegDef_t *pGPIOx, uint8_t PinNumber)

  Reads a value from a specific GPIO pin.
- uint16_t GPIO_ReadFromInputPort (GPIO_RegDef_t *pGPIOx)

  Reads a value from the entire GPIO port.
- void GPIO_WriteToOutputPin (GPIO_RegDef_t *pGPIOx, uint16_t PinNumber, uint8_t value)

  Writes a value to a specific GPIO pin.
- void GPIO_WriteToOutputPort (GPIO_RegDef_t *pGPIOx, uint16_t value)

  Writes a value to the entire GPIO port.
- void GPIO_ToggleOutputPin (GPIO_RegDef_t *pGPIOx, uint16_t PinNumber)

Toggles the output value of a specific GPIO pin.

- void GPIO_IRQInterruptConfig (uint8_t IRQNumber, uint8_t EnorDi)

  Configures the IRQ for a specific GPIO pin.
- void GPIO_IRQPriorityConfig (uint8_t IRQNumber, uint32_t IRQpriority)

  Configures the priority for a specific IRQ.
- void GPIO_IRQHandling (uint16_t PinNumber)

  Handles the IRQ for a specific GPIO pin.

6.8.1 Detailed Description

This file contains the GPIO driver API declarations for the STM32F407xx MCU.

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-08-10

Copyright

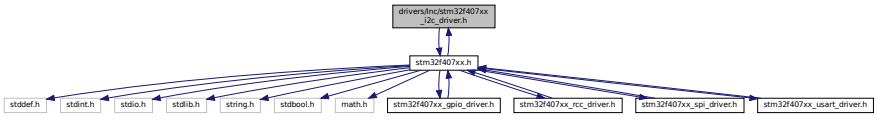
Copyright (c) 2023

6.9 drivers/Inc/stm32f407xx_i2c_driver.h File Reference

This file contains definitions and functions prototypes for the STM32F407xx I2C driver.

```
#include "stm32f407xx.h"
```

Include dependency graph for stm32f407xx_i2c_driver.h:



This graph shows which files directly or indirectly include this file:

Classes

- struct I2C_Config_t

  Configuration structure for I2C peripheral.
- struct I2C_Handle_t

  Handle structure for I2C peripheral.

Macros

- #define I2C_READY 0
- #define I2C_BUSY_IN_RX 1
- #define I2C_BUSY_IN_TX 2
- #define I2C_SC_SPEED_SM 100000
- #define I2C_SC_SPEED_FM4K 400000
- #define I2C_SC_SPEED_FM2K 200000
- #define I2C_ACK_ENABLE 1
- #define I2C_ACK_DISABLE 0
- #define I2C_FM_DUTY_2 0
- #define I2C_FM_DUTY_16_9 1
- #define I2C_FLAG_SB (1 << I2C_SR1_SB)
- #define I2C_FLAG_ADDR (1 << I2C_SR1_ADDR)
- #define I2C_FLAG_BTF (1 << I2C_SR1_BTF)
- #define I2C_FLAG_ADD10 (1 << I2C_SR1_ADD10)
- #define I2C_FLAG_STOPF (1 << I2C_SR1_STOPF)
- #define I2C_FLAG_RxNE (1 << I2C_SR1_RxNE)
- #define I2C_FLAG_TxE (1 << I2C_SR1_TxE)
- #define I2C_FLAG_BERR (1 << I2C_SR1_BERR)
- #define I2C_FLAG_ARLO (1 << I2C_SR1_ARLO)
- #define I2C_FLAG_AF (1 << I2C_SR1_AF)
- #define I2C_FLAG_OVR (1 << I2C_SR1_OVR)
- #define I2C_FLAG_PECERR (1 << I2C_SR1_PECERR)
- #define I2C_FLAG_TIMEOUT (1 << I2C_SR1_TIMEOUT)
- #define I2C_FLAG_SMBALERT (1 << I2C_SR1_SMBALERT)
- #define I2C_ENABLE_SR SET
- #define I2C_DISABLE_SR RESET
- #define I2C_EV_TX_CMPLT 0
- #define I2C_EV_RX_CMPLT 1
- #define I2C_EV_STOP 2
- #define I2C_ERROR_BERR 3
- #define I2C_ERROR_ARLO 4
- #define I2C_ERROR_AF 5
- #define I2C_ERROR_OVR 6
- #define I2C_ERROR_TIMEOUT 7
- #define I2C_EV_DATA_REQ 8
- #define I2C_EV_DATA_RCV 9

Functions

- void I2C_PeriClockControl (I2C_RegDef_t *pI2Cx, uint8_t EnorDi)

  Controls the peripheral clock of the I2C peripheral.
- void I2C_Init (I2C_Handle_t *pI2CHandle)

  Initializes the I2C peripheral.
- void I2C_DeInit (I2C_RegDef_t *pI2Cx)

  Deinitializes the I2C peripheral.
- void I2C_MasterSendData (I2C_Handle_t *pI2CHandle, uint8_t *pTxBuffer, uint32_t Len, uint8_t SlaveAddr, uint8_t SR)

  Sends data in master mode over the I2C bus.
- void I2C_MasterReceiveData (I2C_Handle_t *pI2CHandle, uint8_t *pRxBuffer, uint8_t Len, uint8_t SlaveAddr, uint8_t SR)

  Receives data in master mode over the I2C bus.
- void I2C_SlaveSendData (I2C_RegDef_t *pI2Cx, uint8_t data)

  Sends a single byte of data in slave mode over the I2C bus.
- uint8_t I2C_SlaveReceiveData (I2C_RegDef_t *pI2Cx)

  Receives a single byte of data in slave mode over the I2C bus.
- uint8_t I2C_MasterSendDataIT (I2C_Handle_t *pI2CHandle, uint8_t *pTxBuffer, uint32_t Len, uint8_t SlaveAddr, uint8_t SR)

  Sends data in master mode over the I2C bus with interrupt support.
- uint8_t I2C_MasterReceiveDataIT (I2C_Handle_t *pI2CHandle, uint8_t *pRxBuffer, uint8_t Len, uint8_t SlaveAddr, uint8_t SR)

  Receives data in master mode over the I2C bus with interrupt support.
- void I2C_CloseSendData (I2C_Handle_t *pI2CHandle)

  Closes the transmission process.
- void I2C_CloseReceiveData (I2C_Handle_t *pI2CHandle)

  Closes the reception process.
- void I2C_IRQInterruptConfig (uint8_t IRQNumber, uint8_t EnorDi)

  Configures IRQ interrupt for the I2C peripheral.
- void I2C_IRQPriorityConfig (uint8_t IRQNumber, uint32_t IRQPriority)

  Configures IRQ priority for the I2C peripheral.
- void I2C_EV_IRQHandling (I2C_Handle_t *pI2CHandle)

  Handles I2C event interrupt.
- void I2C_ER_IRQHandling (I2C_Handle_t *pI2CHandle)

  Handles I2C error interrupt.
- void I2C_PeripheralControl (I2C_RegDef_t *pI2Cx, uint8_t EnorDi)

  Controls peripheral operation in master mode.
- uint8_t I2C_GetFlagStatus (I2C_RegDef_t *pI2Cx, uint32_t FlagName)

  Retrieves the flag status of the specified I2C flag.
- void I2C_ManageAcking (I2C_RegDef_t *pI2Cx, uint8_t EnorDi)

  Manages ACKing during I2C communication.
- void I2C_GenerateStopCondition (I2C_RegDef_t *pI2Cx)

  Generates a stop condition on the I2C bus.
- void I2C_SlaveEnableDisableCallbackEvents (I2C_RegDef_t *pI2Cx, uint8_t EnorDi)

  Enables or disables callback events for the I2C slave.
- void I2C_ApplicationEventCallback (I2C_Handle_t *pI2CHandle, uint8_t AppEv)

  Handles application events for the I2C communication.

6.9.1 Detailed Description

This file contains definitions and functions prototypes for the STM32F407xx I2C driver.

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

Copyright

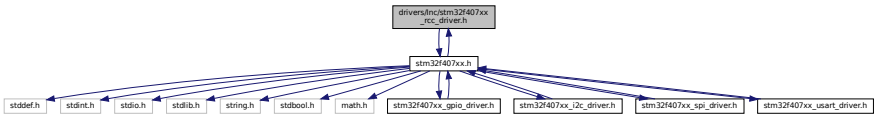
Copyright (c) 2023

6.10 drivers/Inc/stm32f407xx_rcc_driver.h File Reference

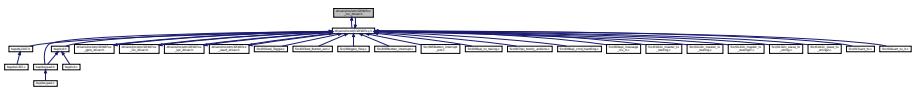
This file contains definitions and functions prototypes for the STM32F407xx SPI driver.

```
#include "stm32f407xx.h"
```

Include dependency graph for stm32f407xx_rcc_driver.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **RCC_PLLCFGR_PLLSRC_HSE** ((uint32_t)0x00400000)
- #define **HSI_VALUE** 16000000U
- #define **HSE_VALUE** 8000000U

Functions

- uint32_t RCC_GetPCLK1Value (void)

  Get the frequency of the PCLK1 (Peripheral Clock 1).
- uint32_t RCC_GetPCLK2Value (void)

  Get the frequency of the PCLK2 (Peripheral Clock 2).
- uint32_t RCC_GetPLLOutputClock (void)

  Get the frequency of the PLL (Phase-Locked Loop) output clock.

6.10.1 Detailed Description

This file contains definitions and functions prototypes for the STM32F407xx SPI driver.

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

Copyright

6.10.2 Function Documentation

6.10.2.1 RCC_GetPCLK1Value()

```
uint32_t RCC_GetPCLK1Value (
            void  )
```

Get the frequency of the PCLK1 (Peripheral Clock 1).

This function calculates and returns the frequency of the PCLK1, which is the peripheral clock for APB1 peripherals.

Returns

The PCLK1 frequency in Hertz.

6.10.2.2 RCC_GetPCLK2Value()

```
uint32_t RCC_GetPCLK2Value (
            void  )
```

Get the frequency of the PCLK2 (Peripheral Clock 2).

This function calculates and returns the frequency of the PCLK2, which is the peripheral clock for APB2 peripherals.

Returns

The PCLK2 frequency in Hertz.

6.10.2.3 RCC_GetPLLOutputClock()

```
uint32_t RCC_GetPLLOutputClock (
            void  )
```

Get the frequency of the PLL (Phase-Locked Loop) output clock.

This function calculates and returns the frequency of the PLL output clock.

Returns

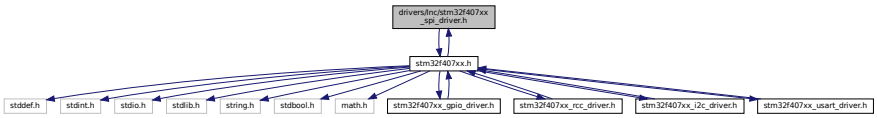
The PLL output clock frequency in Hertz.

6.11 drivers/Inc/stm32f407xx_spi_driver.h File Reference

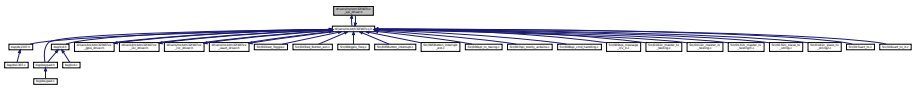
This file contains definitions and functions prototypes for the STM32F407xx SPI driver.

```
#include "stm32f407xx.h"
```

Include dependency graph for stm32f407xx_spi_driver.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct SPI_Config_t

  Configuration structure for SPI peripheral.
- struct SPI_Handle_t

  Handle structure for SPI peripheral.

Macros

- #define SPI_DEVICE_MODE_MASTER 1
- #define SPI_DEVICE_MODE_SLAVE 0
- #define SPI_BUS_CONFIG_FULL_DUPLEX 1
- #define SPI_BUS_CONFIG_HALF_DUPLEX 2
- #define SPI_BUS_CONFIG_SIMPLEX_RX_ONLY 3
- #define SPI_SCLK_SPEED_DIV2 0
- #define SPI_SCLK_SPEED_DIV4 1
- #define SPI_SCLK_SPEED_DIV8 2
- #define SPI_SCLK_SPEED_DIV16 3
- #define SPI_SCLK_SPEED_DIV32 4
- #define SPI_SCLK_SPEED_DIV64 5
- #define SPI_SCLK_SPEED_DIV128 6
- #define SPI_SCLK_SPEED_DIV256 7
- #define SPI_DFF_8BITS 0
- #define SPI_DFF_16BITS 1
- #define SPI_CPOL_HIGH 1
- #define SPI_CPOL_LOW 0
- #define SPI_CPHA_HIGH 1
- #define SPI_CPHA_LOW 0
- #define SPI_SSM_EN 1
- #define SPI_SSM_DI 0
- #define SPI_RXNE_FLAG (1 << SPI_SR_RXNE)
- #define SPI_TXE_FLAG (1 << SPI_SR_TXE)
- #define SPI_CHSIDE_FLAG (1 << SPI_SR_CHSIDE)
- #define SPI_UDR_FLAG (1 << SPI_SR_UDR)
- #define SPI_CRCERR_FLAG (1 << SPI_SR_CRCERR)
- #define SPI_MODF_FLAG (1 << SPI_SR_MODF)
- #define SPI_OVR_FLAG (1 << SPI_SR_OVR)
- #define SPI_BUSY_FLAG (1 << SPI_SR_BSY)
- #define SPI_FRE_FLAG (1 << SPI_SR_FRE)
- #define SPI_READY 0
- #define SPI_BUSY_IN_RX 1
- #define SPI_BUSY_IN_TX 2
- #define SPI_EVENT_TX_CMPLT 1
- #define SPI_EVENT_RX_CMPLT 2
- #define SPI_EVENT_OVR_ERR 3
- #define SPI_EVENT_CRC_ERR 4
- #define CMD_LED_CTRL 0x50
- #define CMD_SENSOR_READ 0x51
- #define CMD_LED_READ 0x52
- #define CMD_PRINT 0x53
- #define CMD_ID_READ 0x54
- #define LED_ON 1
- #define LED_OFF 0

- #define LED_PIN 9
- #define ANALOG_PIN0 0
- #define ANALOG_PIN1 1
- #define ANALOG_PIN2 2
- #define ANALOG_PIN3 3
- #define ANALOG_PIN4 4

Functions

- void SPI_PeripheralClockControl (SPI_RegDef_t *pSPIx, uint8_t EnorDi)

  Enables or disables the peripheral clock for the SPI port.
- void SPI_PeripheralControl (SPI_RegDef_t *pSPIx, uint8_t EnorDi)

  Enables or disables the SPI peripheral.
- void SPI_Init (SPI_Handle_t *pSPIHandle)

  Initializes the SPI port pin according to the configuration.
- void SPI_DeInit (SPI_RegDef_t *pSPIx)

  Deinitializes the SPI port.
- uint8_t SPI_GetFlagStatus (SPI_RegDef_t *pSPIx, uint8_t FlagName)

  Get the status of a specific SPI flag.
- void SPI_SendData (SPI_RegDef_t *pSPIx, uint8_t *pTxBuffer, uint32_t Len)

  Sends data over SPI.
- void SPI_ReceiveData (SPI_RegDef_t *pSPIx, uint8_t *pRxBuffer, uint32_t Len)

  Receives data over SPI.
- uint8_t SPI_SendDataIT (SPI_Handle_t *pSPIHandle, uint8_t *pTxBuffer, uint32_t Len)

  Sends data over SPI using interrupt-driven communication.
- uint8_t SPI_ReceiveDataIT (SPI_Handle_t *pSPIHandle, uint8_t *pRxBuffer, uint32_t Len)

  Receives data over SPI using interrupt-driven communication.
- void SPI_IRQinterruptConfig (uint8_t IRQNumber, uint8_t EnorDi)

  Configures the IRQ for a specific SPI pin.
- void SPI_IRQperiorityConfig (uint8_t IRQNumber, uint32_t IRQpriority)

  Configures the priority for a specific IRQ.
- void SPI_IRQHandling (SPI_Handle_t *pSPIHandle)

  Handles the IRQ for a specific SPI pin.
- void SPI_SSIConfig (SPI_RegDef_t *pSPIx, uint8_t EnorDi)

  Enables or disables the Software Slave Management (SSI) configuration for the SPI peripheral.
- void SPI_SSOEConfig (SPI_RegDef_t *pSPIx, uint8_t EnorDi)

  Enables or disables the SPI Slave Select Output Enable (SSOE) configuration for the SPI peripheral.
- void SPI_ClearOVRFlag (SPI_RegDef_t *pSPIx)

  Clears the overrun error (OVR) flag in the SPI peripheral.
- void SPI_CloseTransmission (SPI_Handle_t *pSPIHandle)

  Closes the transmission operation on the SPI peripheral.
- void SPI_CloseReception (SPI_Handle_t *pSPIHandle)

  Closes the reception operation on the SPI peripheral.
- void SPI_ApplicationEventCallback (SPI_Handle_t *pSPIHandle, uint8_t AppEv)

  SPI Application Event Callback.

6.11.1 Detailed Description

This file contains definitions and functions prototypes for the STM32F407xx SPI driver.

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

Copyright

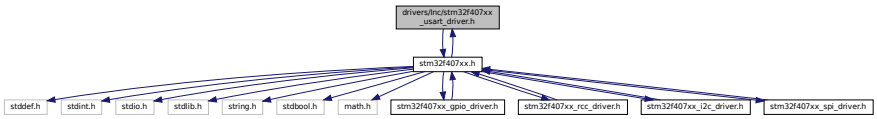
Copyright (c) 2023

6.12 drivers/Inc/stm32f407xx_usart_driver.h File Reference

This file contains definitions and functions prototypes for the STM32F407xx USART driver.

```
#include "stm32f407xx.h"
```

Include dependency graph for stm32f407xx_usart_driver.h:



This graph shows which files directly or indirectly include this file:

Classes

- struct USART_Config_t

  Configuration structure for USART (Universal Synchronous Asynchronous Receiver Transmitter) peripheral.
- struct USART_Handle_t

  Handle structure for USART peripheral.

Macros

- #define USART_MODE_ONLY_TX 0
- #define USART_MODE_ONLY_RX 1
- #define USART_MODE_TXRX 2
- #define **USART_STD_BAUD_1200** (uint8_t)1200
- #define **USART_STD_BAUD_2400** (uint8_t)2400
- #define **USART_STD_BAUD_9600** (uint8_t)9600
- #define **USART_STD_BAUD_19200** (uint8_t)19200
- #define **USART_STD_BAUD_38400** (uint8_t)38400
- #define **USART_STD_BAUD_57600** (uint8_t)57600
- #define **USART_STD_BAUD_115200** (uint8_t)115200
- #define **USART_STD_BAUD_230400** (uint8_t)230400
- #define **USART_STD_BAUD_460800** (uint8_t)460800
- #define **USART_STD_BAUD_921600** (uint8_t)921600
- #define **USART_STD_BAUD_2M** (uint8_t)2000000
- #define **USART_STD_BAUD_3M** (uint8_t)3000000
- #define USART_PARITY_EN_ODD 2
- #define USART_PARITY_EN_EVEN 1
- #define USART_PARITY_DISABLE 0
- #define USART_WORDLEN_8BITS 0
- #define USART_WORDLEN_9BITS 1
- #define USART_STOPBITS_1 0
- #define USART_STOPBITS_0_5 1
- #define USART_STOPBITS_2 2
- #define USART_STOPBITS_1_5 3
- #define USART_HW_FLOW_CTRL_NONE 0
- #define USART_HW_FLOW_CTRL_CTS 1
- #define USART_HW_FLOW_CTRL_RTS 2
- #define USART_HW_FLOW_CTRL_CTS_RTS 3
- #define USART_FLAG_PE (1 << USART_SR_PE)
- #define USART_FLAG_FE (1 << USART_SR_FE)
- #define USART_FLAG_NE (1 << USART_SR_NE)
- #define USART_FLAG_ORE (1 << USART_SR_ORE)
- #define USART_FLAG_IDLE (1 << USART_SR_IDLE)
- #define USART_FLAG_RXNE (1 << USART_SR_RXNE)
- #define USART_FLAG_TC (1 << USART_SR_TC)
- #define USART_FLAG_TXE (1 << USART_SR_TXE)
- #define USART_READY 0
- #define USART_BUSY_IN_RX 1
- #define USART_BUSY_IN_TX 2
- #define USART_EVENT_TX_CMPLT 0
- #define USART_EVENT_RX_CMPLT 1
- #define USART_EVENT_IDLE 2
- #define USART_EVENT_CTS 3
- #define USART_EVENT_PE 4
- #define USART_ERR_FE 5
- #define USART_ERR_NE 6
- #define USART_ERR_ORE 7

Functions

- void USART_PeripheralClockControl (USART_RegDef_t *pUSARTx, uint8_t EnorDi)

  Enable or disable the peripheral clock for the given USARTx.
- void USART_SetBaudRate (USART_RegDef_t *pUSARTx, uint32_t BaudRate)

  Set the baud rate for a USART peripheral.
- void USART_Init (USART_Handle_t *pUSARTHandle)

  Initialize the USART peripheral.
- void USART_DeInit (USART_RegDef_t *pUSARTx)

  Deinitialize the USART peripheral.
- void USART_SendData (USART_Handle_t *pUSARTHandle, uint8_t *pTxBuffer, uint32_t Len)

  Send data over USART.
- void USART_ReceiveData (USART_Handle_t *pUSARTHandle, uint8_t *pRxBuffer, uint32_t Len)

  Receive data from USART.
- uint8_t USART_SendDataIT (USART_Handle_t *pUSARTHandle, uint8_t *pTxBuffer, uint32_t Len)

  Send data over USART using interrupt-driven communication.
- uint8_t USART_ReceiveDataIT (USART_Handle_t *pUSARTHandle, uint8_t *pRxBuffer, uint32_t Len)

  Receive data from USART using interrupt-driven communication.
- void USART_PeripheralControl (USART_RegDef_t *pUSARTx, uint8_t EnorDi)

  Control the USART peripheral (ENABLE/DISABLE).
- uint8_t USART_GetFlagStatus (USART_RegDef_t *pUSARTx, uint8_t FlagName)

  Get the status of a specific USART flag.
- void USART_ClearFlag (USART_RegDef_t *pUSARTx, uint16_t FlagName)

  Clear a specific USART flag.
- void USART_CloseTransmission (USART_Handle_t *pUSARTHandle)

  Close USART transmission.
- void USART_CloseReception (USART_Handle_t *pUSARTHandle)

  Close USART reception.
- void USART_IRQInterruptConfig (uint8_t IRQNumber, uint8_t EnorDi)

  Configure IRQ number and enable/disable IRQ.
- void USART_IRQPriorityConfig (uint8_t IRQNumber, uint32_t IRQPriority)

  Set the priority of an IRQ.
- void USART_IRQHandling (USART_Handle_t *pUSARTHandle)

  Handle USART interrupts.
- void USART_ApplicationEventCallback (USART_Handle_t *pUSARTHandle, uint8_t AppEv)

  Application callback function for USART events.
- void USART_ClearEventErrFlag (USART_RegDef_t *pUSARTx)

  Clears the error flags in the USART status register.

6.12.1 Detailed Description

This file contains definitions and functions prototypes for the STM32F407xx USART driver.

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

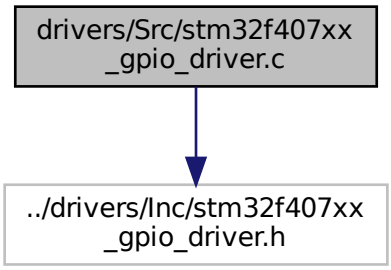
Copyright

6.13 drivers/Src/stm32f407xx_gpio_driver.c File Reference

This file contains the GPIO driver implementation.

```
#include "../drivers/Inc/stm32f407xx_gpio_driver.h"
```

Include dependency graph for stm32f407xx_gpio_driver.c:



Functions

- void GPIO_PeripheralClockControl (GPIO_RegDef_t *pGPIOx, uint8_t EnorDi)
  This function enables or disables the peripheral clock for a given GPIO port.
- void GPIO_Init (GPIO_Handle_t *pGPIOHandle)
  Initialize the GPIO port.
- void GPIO_DeInit (GPIO_RegDef_t *pGPIOx)
  Deinitialize the GPIO port.
- uint8_t GPIO_ReadFromInputPin (GPIO_RegDef_t *pGPIOx, uint8_t PinNumber)
  Read from a specific GPIO pin.
- uint16_t GPIO_ReadFromInputPort (GPIO_RegDef_t *pGPIOx)
  Read from all pins of a GPIO port.
- void GPIO_WriteToOutputPin (GPIO_RegDef_t *pGPIOx, uint16_t PinNumber, uint8_t value)
  Write to a specific GPIO pin.
- void GPIO_WriteToOutputPort (GPIO_RegDef_t *pGPIOx, uint16_t value)
  Write to all pins of a GPIO port.
- void GPIO_ToggleOutputPin (GPIO_RegDef_t *pGPIOx, uint16_t PinNumber)
  Toggle a specific GPIO pin.
- void GPIO_IRQInterruptConfig (uint8_t IRQNumber, uint8_t EnorDi)
  Configure the interrupt for a specific GPIO pin.
- void GPIO_IRQPriorityConfig (uint8_t IRQNumber, uint32_t IRQPriority)
  Configure the priority of a specific IRQ.
- void GPIO_IRQHandling (uint16_t PinNumber)
  Handle the interrupt for a specific GPIO pin.

6.13.1 Detailed Description

This file contains the GPIO driver implementation.

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

Copyright

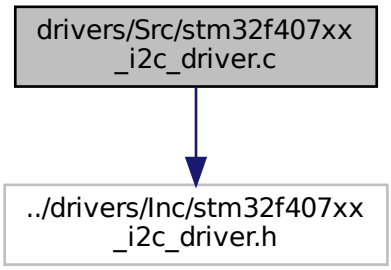
Copyright (c) 2023

6.14 drivers/Src/stm32f407xx_i2c_driver.c File Reference

This file contains the implementation of the I2C driver APIs.

```
#include "../drivers/Inc/stm32f407xx_i2c_driver.h"
```

Include dependency graph for stm32f407xx_i2c_driver.c:



Functions

- void I2C_GenerateStopCondition (I2C_RegDef_t *pI2Cx)

  Generates the stop condition on the I2C bus.
- void I2C_PeriClockControl (I2C_RegDef_t *pI2Cx, uint8_t EnorDi)

  Enables or disables the peripheral clock for the given I2C peripheral.
- void I2C_Init (I2C_Handle_t *pI2CHandle)

  Initializes the I2C peripheral with the provided configuration settings.
- void I2C_DeInit (I2C_RegDef_t *pI2Cx)

  Deinitializes the I2C peripheral.
- void I2C_MasterSendData (I2C_Handle_t *pI2CHandle, uint8_t *pTxBuffer, uint32_t Len, uint8_t SlaveAddr, uint8_t SR)

  Sends data over I2C as a master to the specified slave device.
- void I2C_MasterReceiveData (I2C_Handle_t *pI2CHandle, uint8_t *pRxBuffer, uint8_t Len, uint8_t SlaveAddr, uint8_t SR)

  Receives data over I2C as a master from the specified slave device.
- void I2C_IRQInterruptConfig (uint8_t IRQNumber, uint8_t EnorDi)

  Configures the interrupt for the specified IRQ number.
- void I2C_IRQPriorityConfig (uint8_t IRQNumber, uint32_t IRQPriority)

  Configures the priority for the specified IRQ number.
- uint8_t I2C_MasterSendDataIT (I2C_Handle_t *pI2CHandle, uint8_t *pTxBuffer, uint32_t Len, uint8_t SlaveAddr, uint8_t Sr)

  Sends data over I2C as a master using interrupt-driven communication.
- uint8_t I2C_MasterReceiveDataIT (I2C_Handle_t *pI2CHandle, uint8_t *pRxBuffer, uint8_t Len, uint8_t SlaveAddr, uint8_t Sr)

  Sends data over I2C as a master using interrupt-driven communication.
- void I2C_EV_IRQHandling (I2C_Handle_t *pI2CHandle)

  Handles I2C event interrupts and calls appropriate event callback functions.
- void I2C_CloseSendData (I2C_Handle_t *pI2CHandle)

  Closes the I2C data transmission, disables relevant interrupts, and resets handle members.
- void I2C_CloseReceiveData (I2C_Handle_t *pI2CHandle)

  Closes the I2C data reception in slave mode, disables relevant interrupts, and resets handle members.
- void I2C_SlaveSendData (I2C_RegDef_t *pI2Cx, uint8_t data)

  Sends data in slave mode.
- uint8_t I2C_SlaveReceiveData (I2C_RegDef_t *pI2Cx)

  Receives data in slave mode.
- void I2C_ER_IRQHandling (I2C_Handle_t *pI2CHandle)

  Handles I2C error interrupts and calls appropriate error callback functions.
- void I2C_PeripheralControl (I2C_RegDef_t *pI2Cx, uint8_t EnorDi)

  Enables or disables the I2C peripheral.
- uint8_t I2C_GetFlagStatus (I2C_RegDef_t *pI2Cx, uint32_t FlagName)

  Gets the status of a specific flag in the I2C status register.
- void I2C_ManageAcking (I2C_RegDef_t *pI2Cx, uint8_t EnorDi)

  Manages acknowledgment control in I2C communication.
- void I2C_SlaveEnableDisableCallbackEvents (I2C_RegDef_t *pI2Cx, uint8_t EnorDi)

  Enables or disables callback events for the I2C slave.
- __weak void I2C_ApplicationEventCallback (I2C_Handle_t *pI2CHandle, uint8_t AppEv)

  Application callback function for I2C events.

6.14.1 Detailed Description

This file contains the implementation of the I2C driver APIs.

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

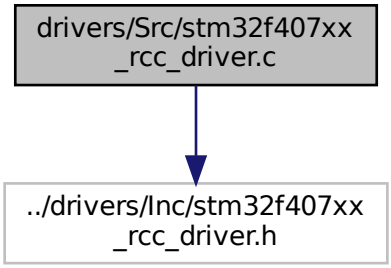
Copyright

6.15 drivers/Src/stm32f407xx_rcc_driver.c File Reference

This file contains the RCC (Reset and Clock Control) driver implementation for STM32F407xx microcontrollers.

```
#include "../drivers/Inc/stm32f407xx_rcc_driver.h"
```

Include dependency graph for stm32f407xx_rcc_driver.c:



Functions

- uint32_t RCC_GetPCLK1Value (void)

  Get the frequency of the PCLK1 (Peripheral Clock 1).
- uint32_t RCC_GetPCLK2Value (void)

  Get the frequency of the PCLK2 (Peripheral Clock 2).
- uint32_t RCC_GetPLLOutputClock (void)

  Get the frequency of the PLL (Phase-Locked Loop) output clock.

Variables

- uint16_t **AHB_PreScaler** [8] = {2, 4, 8, 16, 64, 128, 256, 512}
- uint8_t **APB1_PreScaler** [4] = {2, 4, 8, 16}

6.15.1 Detailed Description

This file contains the RCC (Reset and Clock Control) driver implementation for STM32F407xx microcontrollers.

Author

Mohamed Ali Haoufa

This driver provides functions for configuring and controlling the system clock and peripheral clocks.

Version

0.1

Date

2023-10-07

Copyright

6.15.2 Function Documentation

6.15.2.1 RCC_GetPCLK1Value()

```
uint32_t RCC_GetPCLK1Value (
            void  )
```

Get the frequency of the PCLK1 (Peripheral Clock 1).

This function calculates and returns the frequency of the PCLK1, which is the peripheral clock for APB1 peripherals.

Returns

The PCLK1 frequency in Hertz.

6.15.2.2 RCC_GetPCLK2Value()

```
uint32_t RCC_GetPCLK2Value (
            void  )
```

Get the frequency of the PCLK2 (Peripheral Clock 2).

This function calculates and returns the frequency of the PCLK2, which is the peripheral clock for APB2 peripherals.

Returns

The PCLK2 frequency in Hertz.

6.15.2.3 RCC_GetPLLOutputClock()

```
uint32_t RCC_GetPLLOutputClock (
            void  )
```

Get the frequency of the PLL (Phase-Locked Loop) output clock.

This function calculates and returns the frequency of the PLL output clock.

Returns

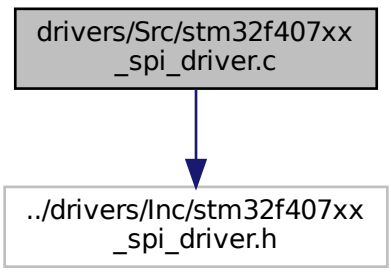
The PLL output clock frequency in Hertz.

6.16 drivers/Src/stm32f407xx_spi_driver.c File Reference

This file contains the implementation of the SPI driver APIs.

```
#include "../drivers/Inc/stm32f407xx_spi_driver.h"
```

Include dependency graph for stm32f407xx_spi_driver.c:



Functions

- void SPI_PeripheralClockControl (SPI_RegDef_t *pSPIx, uint8_t EnorDi)

  Enables or disables the peripheral clock for the SPI port.
- void SPI_PeripheralControl (SPI_RegDef_t *pSPIx, uint8_t EnorDi)

  Enables or disables the SPI peripheral.
- void SPI_Init (SPI_Handle_t *pSPIHandle)

  Initializes the SPI port pin according to the configuration.
- void SPI_DeInit (SPI_RegDef_t *pSPIx)

  Deinitializes the SPI port.
- uint8_t SPI_GetFlagStatus (SPI_RegDef_t *pSPIx, uint8_t FlagName)

  Get the status of a specific SPI flag.
- void SPI_SendData (SPI_RegDef_t *pSPIx, uint8_t *pTxBuffer, uint32_t Len)

  Sends data over SPI. it's a blocking call.
- void SPI_ReceiveData (SPI_RegDef_t *pSPIx, uint8_t *pRxBuffer, uint32_t Len)

  Receives data over SPI.
- void SPI_SSIConfig (SPI_RegDef_t *pSPIx, uint8_t EnorDi)

  Enables or disables the Software Slave Management (SSI) configuration for the SPI peripheral.
- void SPI_SSOEConfig (SPI_RegDef_t *pSPIx, uint8_t EnorDi)

  Enables or disables the SPI Slave Select Output Enable (SSOE) configuration for the SPI peripheral.
- void SPI_IRQinterruptConfig (uint8_t IRQNumber, uint8_t EnorDi)

  Configures the IRQ for a specific SPI pin.
- void SPI_IRQperiorityConfig (uint8_t IRQNumber, uint32_t IRQpriority)

  Configures the priority for a specific IRQ.
- void SPI_IRQHandling (SPI_Handle_t *pSPIHandle)

  Handles the IRQ for a specific SPI pin.
- uint8_t SPI_SendDataIT (SPI_Handle_t *pSPIHandle, uint8_t *pTxBuffer, uint32_t Len)

  Sends data over SPI using interrupt-driven communication.
- uint8_t SPI_ReceiveDataIT (SPI_Handle_t *pSPIHandle, uint8_t *pRxBuffer, uint32_t Len)

  Receives data over SPI using interrupt-driven communication.
- void SPI_ClearOVRFlag (SPI_RegDef_t *pSPIx)

  Clears the overrun error (OVR) flag in the SPI peripheral.
- void SPI_CloseTransmission (SPI_Handle_t *pSPIHandle)

  Closes the transmission operation on the SPI peripheral.
- void SPI_CloseReception (SPI_Handle_t *pSPIHandle)

  Closes the reception operation on the SPI peripheral.
- __weak void SPI_ApplicationEventCallback (SPI_Handle_t *pSPIHandle, uint8_t AppEv)

  SPI Application Event Callback.

6.16.1 Detailed Description

This file contains the implementation of the SPI driver APIs.

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

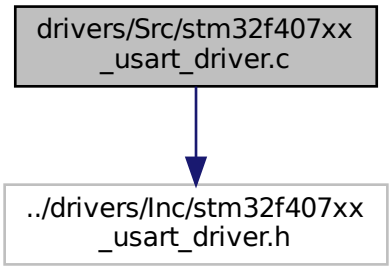
Copyright

6.17 drivers/Src/stm32f407xx_usart_driver.c File Reference

This file contains the implementation of the USART driver APIs.

```
#include "../drivers/Inc/stm32f407xx_usart_driver.h"
```

Include dependency graph for stm32f407xx_usart_driver.c:



Functions

- void USART_PeripheralClockControl (USART_RegDef_t *pUSARTx, uint8_t EnorDi)

  Enable or disable the peripheral clock for the given USARTx.
- void USART_PeripheralControl (USART_RegDef_t *pUSARTx, uint8_t EnorDi)

  Control the USART peripheral (ENABLE/DISABLE).
- void USART_SetBaudRate (USART_RegDef_t *pUSARTx, uint32_t BaudRate)

  Set the baud rate for a USART peripheral.
- void USART_Init (USART_Handle_t *pUSARTHandle)

  Initialize the USART peripheral.
- void USART_DeInit (USART_RegDef_t *pUSARTx)

  Deinitialize the USART peripheral.
- void USART_SendData (USART_Handle_t *pUSARTHandle, uint8_t *pTxBuffer, uint32_t Len)

  Send data over USART.
- void USART_ReceiveData (USART_Handle_t *pUSARTHandle, uint8_t *pRxBuffer, uint32_t Len)

  Receive data from USART.
- uint8_t USART_SendDataIT (USART_Handle_t *pUSARTHandle, uint8_t *pTxBuffer, uint32_t Len)

  Send data over USART using interrupt-driven communication.
- uint8_t USART_ReceiveDataIT (USART_Handle_t *pUSARTHandle, uint8_t *pRxBuffer, uint32_t Len)

  Receive data from USART using interrupt-driven communication.

- uint8_t USART_GetFlagStatus (USART_RegDef_t *pUSARTx, uint8_t FlagName)

  Get the status of a specific USART flag.
- void USART_ClearFlag (USART_RegDef_t *pUSARTx, uint16_t FlagName)

  Clear a specific USART flag.
- void USART_IRQInterruptConfig (uint8_t IRQNumber, uint8_t EnorDi)

  Configure IRQ number and enable/disable IRQ.
- void USART_IRQPriorityConfig (uint8_t IRQNumber, uint32_t IRQPriority)

  Set the priority of an IRQ.
- void USART_IRQHandling (USART_Handle_t *pUSARTHandle)

  Handle USART interrupts.
- __weak void USART_ApplicationEventCallback (USART_Handle_t *pUSARTHandle, uint8_t AppEv)

  Application callback function for USART events.
- void USART_ClearEventErrFlag (USART_RegDef_t *pUSARTx)

  Clears the event/error flags in the USART status register.
- void USART_CloseTransmission (USART_Handle_t *pUSARTHandle)

  Close USART transmission.
- void USART_CloseReception (USART_Handle_t *pUSARTHandle)

  Close USART reception.

6.17.1 Detailed Description

This file contains the implementation of the USART driver APIs.

Author

Mohamed Ali Haoufa

Version

0.1

Date

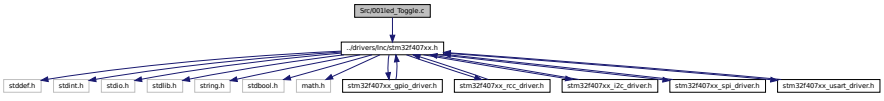
2023-10-07

Copyright

6.18 Src/001led_Toggle.c File Reference

```
#include "../drivers/Inc/stm32f407xx.h"
```

Include dependency graph for 001led_Toggle.c:



Functions

- int main (void)

6.18.1 Detailed Description

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

Copyright

Copyright (c) 2023

6.18.2 Function Documentation

6.18.2.1 main()

```
int main (
            void  )
```

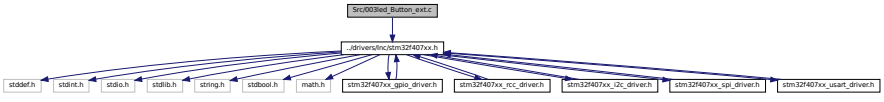
Returns

int

6.19 Src/003led_Button_ext.c File Reference

```
#include "../drivers/Inc/stm32f407xx.h"
```

Include dependency graph for 003led_Button_ext.c:



Macros

- #define **HIGH** 0x1
- #define **BTN_PRESSED** HIGH

Functions

- void **delay** (void)
- int **main** (void)

Variables

- uint8_t volatile **g_button_pressed** = 0

6.19.1 Detailed Description

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

Copyright

Copyright (c) 2023

6.20 Src/004gpio_freq.c File Reference

```
#include "../drivers/Inc/stm32f407xx.h"
```

Include dependency graph for 004gpio_freq.c:

Macros

- #define **HIGH** 1
- #define **LOW** 0
- #define **BTN_PRESSED** LOW

Functions

- int **main** (void)

6.20.1 Detailed Description

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

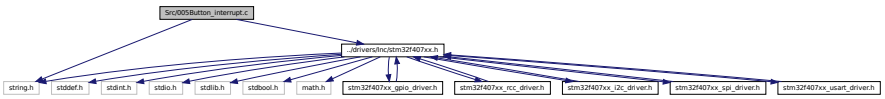
Copyright

Copyright (c) 2023

6.21 Src/005Button_interrupt.c File Reference

```
#include "../drivers/Inc/stm32f407xx.h"
#include <string.h>
```

Include dependency graph for 005Button_interrupt.c:



Macros

- #define **HIGH** 1
- #define **LOW** 0
- #define **Btn_pressed** LOW

Functions

- void **EXTI0_IRQHandler** (void)
- int **main** (void)

Variables

- uint8_t volatile **g_button_pressed** = 0
- uint32_t **g_button_pressed_count** = 0

6.21.1 Detailed Description

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

Copyright

Copyright (c) 2023

6.22 Src/005Button_interrupt_ext.c File Reference

```
#include "../drivers/Inc/stm32f407xx.h"
#include <string.h>
```

Include dependency graph for 005Button_interrupt_ext.c:

Macros

- #define **HIGH** 1
- #define **LOW** 0
- #define **Btn_pressed** LOW

Functions

- void **EXTI9_5_IRQHandler** (void)
- int **main** (void)

6.22.1 Detailed Description

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

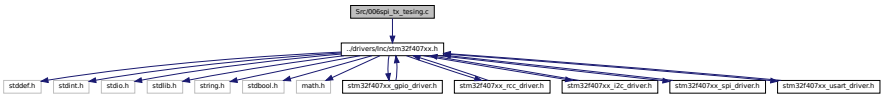
Copyright

Copyright (c) 2023

6.23 Src/006spi_tx_tesing.c File Reference

```
#include "../drivers/Inc/stm32f407xx.h"
```

Include dependency graph for 006spi_tx_tesing.c:



Functions

- void **SPI2_GPIOInits** (void)
- void **SPI2_Inits** (void)
- int **main** (void)

6.23.1 Detailed Description

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

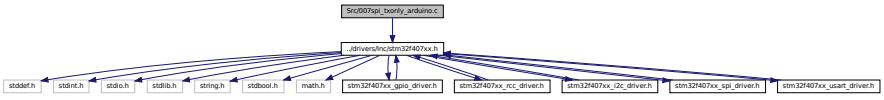
Copyright

Copyright (c) 2023

6.24 Src/007spi_txonly_arduino.c File Reference

```
#include "../drivers/Inc/stm32f407xx.h"
```

Include dependency graph for 007spi_txonly_arduino.c:



Functions

- void **SPI2_GPIOInits** (void)
- void **SPI2_Inits** (void)
- void **GPIO_ButtonInit** (void)
- int **main** (void)

6.24.1 Detailed Description

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

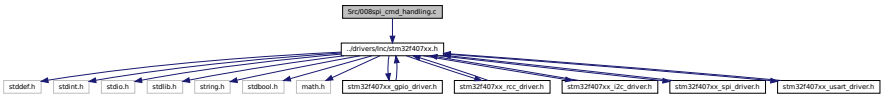
Copyright

Copyright (c) 2023

6.25 Src/008spi_cmd_handling.c File Reference

```
#include "../drivers/Inc/stm32f407xx.h"
```

Include dependency graph for 008spi_cmd_handling.c:



Macros

- #define CMD_LED_CTRL 0x50
- #define CMD_SENSOR_READ 0x51
- #define CMD_LED_READ 0x52
- #define CMD_PRINT 0x53
- #define CMD_ID_READ 0x54
- #define LED_ON 1
- #define LED_OFF 0
- #define LED_PIN 9
- #define ANALOG_PIN0 0
- #define ANALOG_PIN1 1
- #define ANALOG_PIN2 2
- #define ANALOG_PIN3 3
- #define ANALOG_PIN4 4

Functions

- void **SPI2_GPIOInits** (void)
- void **SPI2_Inits** (void)
- void **GPIO_ButtonInit** (void)
- uint8_t **SPI_VerifyResponse** (uint8_t ackbyte)
- int **main** (void)

6.25.1 Detailed Description

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

Copyright

6.25.2 Macro Definition Documentation

6.25.2.1 ANALOG_PIN0

```
#define ANALOG_PIN0 0
```

Represents analog pin 0.

6.25.2.2 ANALOG_PIN1

```
#define ANALOG_PIN1 1
```

Represents analog pin 1.

6.25.2.3 ANALOG_PIN2

```
#define ANALOG_PIN2 2
```

Represents analog pin 2.

6.25.2.4 ANALOG_PIN3

```
#define ANALOG_PIN3 3
```

Represents analog pin 3.

6.25.2.5 ANALOG_PIN4

```
#define ANALOG_PIN4 4
```

Represents analog pin 4.

6.25.2.6 CMD_ID_READ

```
#define CMD_ID_READ 0x54
```

CMD to read an ID.

6.25.2.7 CMD_LED_CTRL

```
#define CMD_LED_CTRL 0x50
```

CMD to control an LED.

6.25.2.8 CMD_LED_READ

```
#define CMD_LED_READ 0x52
```

CMD to read the LED state.

6.25.2.9 CMD_PRINT

```
#define CMD_PRINT 0x53
```

CMD to print data.

6.25.2.10 CMD_SENSOR_READ

```
#define CMD_SENSOR_READ 0x51
```

CMD to read a sensor.

6.25.2.11 LED_OFF

```
#define LED_OFF 0
```

Represents the LED OFF state.

6.25.2.12 LED_ON

```
#define LED_ON 1
```

Represents the LED ON state.

6.25.2.13 LED_PIN

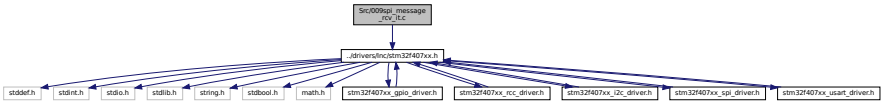
```
#define LED_PIN 9
```

connected the LED to the Arduino pin number 9.

6.26 Src/009spi_message_rcv_it.c File Reference

```
#include "../drivers/Inc/stm32f407xx.h"
```

Include dependency graph for 009spi_message_rcv_it.c:



Macros

- #define **MAX_LEN** 500

Functions

- void **delay** (void)
- void **SPI2_GPIOInits** (void)
- void **SPI2_Inits** (void)
- void **Slave_GPIO_InterruptPinInit** (void)
- int **main** (void)
- void **SPI2_IRQHandler** (void)
- void SPI_ApplicationEventCallback (SPI_Handle_t *pSPIHandle, uint8_t AppEv)

  SPI Application Event Callback.
- void **EXTI9_5_IRQHandler** (void)

Variables

- SPI_Handle_t **SPI2handle**
- char **RcvBuff** [MAX_LEN]
- volatile char **ReadByte**
- volatile uint8_t **rcvStop** = 0
- volatile uint8_t **dataAvailable** = 0

6.26.1 Detailed Description

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

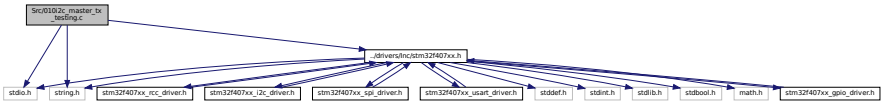
Copyright

Copyright (c) 2023

6.27 Src/010i2c_master_tx_testing.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "../drivers/Inc/stm32f407xx.h"
```

Include dependency graph for 010i2c_master_tx_testing.c:



Macros

- #define **MY_ADDR** 0x61
- #define **SLAVE_ADDR** 0x68

Functions

- void **I2C1_GPIOInits** (void)
- void **I2C1_Inits** (void)
- void **GPIO_ButtonInit** (void)
- int **main** (void)

6.27.1 Detailed Description

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

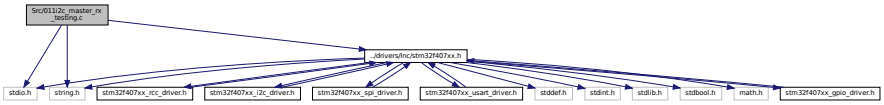
Copyright

Copyright (c) 2023

6.28 Src/011i2c_master_rx_testing.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "../drivers/Inc/stm32f407xx.h"
```

Include dependency graph for 011i2c_master_rx_testing.c:



Macros

- #define **MY_ADDR** 0x61
- #define **SLAVE_ADDR** 0x68

Functions

- void **I2C1_GPIOInits** (void)
- void **I2C1_Inits** (void)
- void **GPIO_ButtonInit** (void)
- int **main** (void)

6.28.1 Detailed Description

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

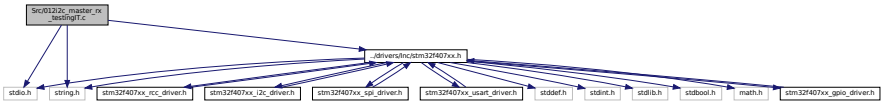
Copyright

Copyright (c) 2023

6.29 Src/012i2c_master_rx_testingIT.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "../drivers/Inc/stm32f407xx.h"
```

Include dependency graph for 012i2c_master_rx_testingIT.c:



Macros

- #define **MY_ADDR** 0x61
- #define **SLAVE_ADDR** 0x68

Functions

- void **I2C1_EV_IRQHandler** (void)
- void **I2C1_ER_IRQHandler** (void)
- void **I2C1_GPIOInits** (void)
- void **I2C1_Inits** (void)
- void **GPIO_ButtonInit** (void)
- int **main** (void)
- void I2C_ApplicationEventCallback (I2C_Handle_t *pI2CHandle, uint8_t AppEv)

  Handles application events for the I2C communication.

6.29.1 Detailed Description

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

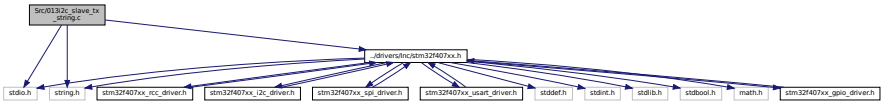
Copyright

Copyright (c) 2023

6.30 Src/013i2c_slave_tx_string.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "../drivers/Inc/stm32f407xx.h"
```

Include dependency graph for 013i2c_slave_tx_string.c:



Macros

- #define **SLAVE_ADDR** 0x68
- #define **MY_ADDR** SLAVE_ADDR

Functions

- void **I2C1_EV_IRQHandler** (void)
- void **I2C1_ER_IRQHandler** (void)
- void **I2C1_GPIOInits** (void)
- void **I2C1_Inits** (void)
- void **GPIO_ButtonInit** (void)
- int **main** (void)
- void I2C_ApplicationEventCallback (I2C_Handle_t *pI2CHandle, uint8_t AppEv)

  Handles application events for the I2C communication.

6.30.1 Detailed Description

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

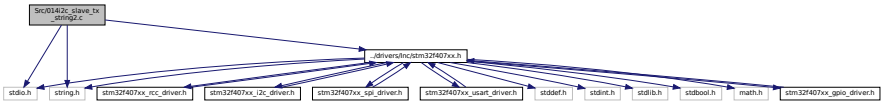
Copyright

Copyright (c) 2023

6.31 Src/014i2c_slave_tx_string2.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "../drivers/Inc/stm32f407xx.h"
```

Include dependency graph for 014i2c_slave_tx_string2.c:



Macros

- #define **SLAVE_ADDR** 0x68
- #define **MY_ADDR** SLAVE_ADDR

Functions

- void **I2C1_EV_IRQHandler** (void)
- void **I2C1_ER_IRQHandler** (void)
- void **I2C1_GPIOInits** (void)
- void **I2C1_Inits** (void)
- void **GPIO_ButtonInit** (void)
- int **main** (void)
- void I2C_ApplicationEventCallback (I2C_Handle_t *pI2CHandle, uint8_t AppEv)

  Handles application events for the I2C communication.

6.31.1 Detailed Description

Author

Mohamed Ali Haoufa

Version

0.1

Date

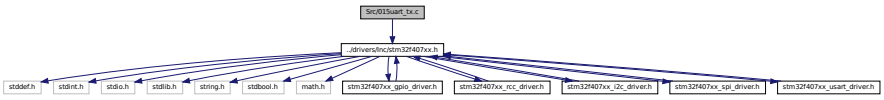
2023-10-07

Copyright

6.32 Src/015uart_tx.c File Reference

```
#include "../drivers/Inc/stm32f407xx.h"
```

Include dependency graph for 015uart_tx.c:



Functions

- void **USART2_Init** (void)
- void **USART2_GPIOInit** (void)
- void **GPIO_ButtonInit** (void)
- void **delay** (void)
- int **main** (void)

Variables

- char **msg** [1024] = "UART Tx testing...\n\r"
- USART_Handle_t **usart2_handle**

6.32.1 Detailed Description

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

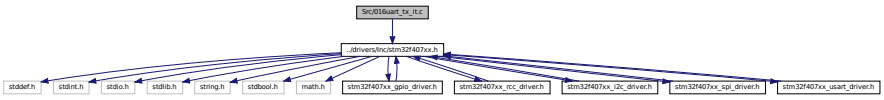
Copyright

Copyright (c) 2023

6.33 Src/016uart_tx_it.c File Reference

```
#include "../drivers/Inc/stm32f407xx.h"
```

Include dependency graph for 016uart_tx_it.c:



Functions

- void **initialise_monitor_handles** ()
- void **USART2_Init** (void)
- void **USART2_GPIOInit** (void)
- void **GPIO_ButtonInit** (void)
- void **delay** (void)
- int **main** (void)
- void **USART2_IRQHandler** (void)
- void USART_ApplicationEventCallback (USART_Handle_t *pUSARTHandle, uint8_t ApEv)

  Application callback function for USART events.

Variables

- char * **msg** [3] = {"hihihihihihi123", "Hello How are you ?" , "Today is Monday !"}
- char **rx_buf** [1024]
- USART_Handle_t **usart2_handle**
- uint8_t **rxCmplt** = RESET
- uint8_t **g_data** = 0

6.33.1 Detailed Description

Author

Mohamed Ali Haoufa

Version

0.1

Date

2023-10-07

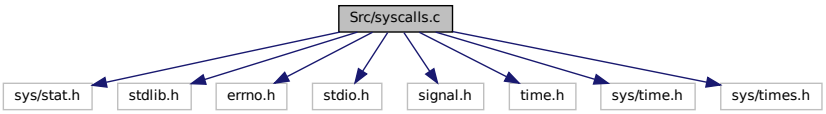
Copyright

6.34 Src/syscalls.c File Reference

STM32CubeIDE Minimal System calls file.

```
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
```

Include dependency graph for syscalls.c:



Macros

- #define **DEMCR** ∗((volatile uint32_t∗) 0xE000EDFCU)
- #define **ITM_STIMULUS_PORT0** ∗((volatile uint32_t∗) 0xE0000000)
- #define **ITM_TRACE_EN** ∗((volatile uint32_t∗) 0xE0000E00)

Functions

- void **ITM_SendChar** (uint8_t ch)
- int **__io_putchar** (int ch) __attribute__((weak))
- int **__io_getchar** (void)
- void **initialise_monitor_handles** ()
- int **_getpid** (void)
- int **_kill** (int pid, int sig)
- void **_exit** (int status)
- **__attribute__** ((weak))
- int **_close** (int file)
- int **_fstat** (int file, struct stat *st)
- int **_isatty** (int file)
- int **_lseek** (int file, int ptr, int dir)
- int **_open** (char *path, int flags,...)
- int **_wait** (int *status)
- int **_unlink** (char *name)
- int **_times** (struct tms *buf)
- int **_stat** (char *file, struct stat *st)
- int **_link** (char *old, char *new)
- int **_fork** (void)
- int **_execve** (char *name, char **argv, char **env)

Variables

- char ** **environ** = __env

6.34.1 Detailed Description

STM32CubeIDE Minimal System calls file.

Author

Auto-generated by STM32CubeIDE

```
For more information about which c-functions
need which of these lowlevel functions
please consult the Newlib libc-manual
```

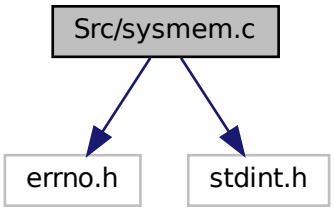
Attention

6.35 Src/sysmem.c File Reference

STM32CubeIDE System Memory calls file.

```
#include <errno.h>
#include <stdint.h>
```

Include dependency graph for sysmem.c:



Functions

- void * _sbrk (ptrdiff_t incr)

  _sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

6.35.1 Detailed Description

STM32CubeIDE System Memory calls file.

Author

Generated by STM32CubeIDE

```
For more information about which C functions
need which of these lowlevel functions
please consult the newlib libc manual
```

Attention

6.35.2 Function Documentation

6.35.2.1 _sbrk()

```
void* _sbrk (
             ptrdiff_t incr )
```

_sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

```
* ##########################################################################
* #  .data  #  .bss  #       newlib heap       #          MSP stack          #
* #         #        #                         # Reserved by _Min_Stack_Size #
* ##########################################################################
* ^-- RAM start      ^-- _end                             _estack, RAM end --^
*
```

This implementation starts allocating at the '_end' linker symbol The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack The implementation considers '_estack' linker symbol to be RAM end NOTE: If the MSP stack, at any point during execution, grows larger than the reserved size, please increase the '_Min_Stack_Size'.

Parameters

incr	Memory size

Returns

Pointer to allocated memory

Index