My Project

Generated by Doxygen 1.10.0

Chapter 1

leds

led	B0	B1	B2
t_north	000ryg00	00000000	00000000
t_east	ryg00000	00000000	00000000
p_north	00000000	000rgb00	00000000
t_south	00000000	ryg00000	00000000
p_west	00000000	00000000	000rgb00
t_west	00000000	00000000	ryg00000

Chapter 2

Topic Index

2.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Topic Documentation

5.1 CMSIS

Topics

- Stm32l4xx_system

5.1.1 Detailed Description

5.1.2 Stm32l4xx_system

Topics

- STM32L4xx_System_Private_Includes
- STM32L4xx_System_Private_TypesDefinitions
- STM32L4xx_System_Private_Defines
- STM32L4xx_System_Private_Macros
- STM32L4xx_System_Private_Variables
- STM32L4xx_System_Private_FunctionPrototypes
- STM32L4xx_System_Private_Functions

5.1.2.1 Detailed Description

5.1.2.2 STM32L4xx_System_Private_Includes

5.1.2.3 STM32L4xx_System_Private_TypesDefinitions

5.1.2.4 STM32L4xx_System_Private_Defines

Macros

- #define HSE_VALUE 8000000U
- #define MSI_VALUE 4000000U
- #define HSI_VALUE 16000000U

5.1.2.4.1 Detailed Description

5.1.2.4.2 Macro Definition Documentation

5.1.2.4.2.1 HSE_VALUE

```
#define HSE_VALUE 8000000U
```

Value of the External oscillator in Hz

5.1.2.4.2.2 HSI_VALUE

```
#define HSI_VALUE 16000000U
```

Value of the Internal oscillator in Hz

5.1.2.4.2.3 MSI_VALUE

```
#define MSI_VALUE 4000000U
```

Value of the Internal oscillator in Hz

5.1.2.5 STM32L4xx_System_Private_Macros

5.1.2.6 STM32L4xx_System_Private_Variables

Variables

- uint32_t **SystemCoreClock** = 4000000U
- const uint8_t **AHBPrescTable** [16] = {0U, 0U, 0U, 0U, 0U, 0U, 0U, 0U, 1U, 2U, 3U, 4U, 6U, 7U, 8U, 9U}
- const uint8_t **APBPrescTable** [8] = {0U, 0U, 0U, 0U, 1U, 2U, 3U, 4U}
- const uint32_t MSIRangeTable [12]

5.1.2.6.1 Detailed Description

5.1.2.6.2 Variable Documentation

5.1.2.6.2.1 MSIRangeTable

```
const uint32_t MSIRangeTable[12]
```

Initial value:

```
= {100000U,   200000U,   400000U,   800000U,  1000000U,  2000000U,
                                    4000000U, 8000000U, 16000000U, 24000000U, 32000000U, 48000000U}
```

5.1.2.7 STM32L4xx_System_Private_FunctionPrototypes

5.1.2.8 STM32L4xx_System_Private_Functions

Functions

- void SystemInit (void)

  Setup the microcontroller system.
- void SystemCoreClockUpdate (void)

  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

5.1.2.8.1 Detailed Description

5.1.2.8.2 Function Documentation

5.1.2.8.2.1 SystemCoreClockUpdate()

```
void SystemCoreClockUpdate (
            void  )
```

Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Note

Each time the core clock (HCLK) changes, this function must be called to update SystemCoreClock variable value. Otherwise, any configuration based on this variable will be incorrect.

- The system frequency computed by this function is not the real frequency in the chip. It is calculated based on the predefined constant and the selected clock source:

- If SYSCLK source is MSI, SystemCoreClock will contain the MSI_VALUE(∗)
- If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(∗∗)
- If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(∗∗∗)
- If SYSCLK source is PLL, SystemCoreClock will contain the HSE_VALUE(∗∗∗) or HSI_VALUE(∗) or MSI_VALUE(∗) multiplied/divided by the PLL factors.

(∗) MSI_VALUE is a constant defined in stm32l4xx_hal.h file (default value 4 MHz) but the real value may vary depending on the variations in voltage and temperature.

(∗∗) HSI_VALUE is a constant defined in stm32l4xx_hal.h file (default value 16 MHz) but the real value may vary depending on the variations in voltage and temperature.

(∗∗∗) HSE_VALUE is a constant defined in stm32l4xx_hal.h file (default value 8 MHz), user has to ensure that HSE_VALUE is same as the real frequency of the crystal used. Otherwise, this function may have wrong result.

- The result of this function could be not correct when using fractional value for HSE crystal.

Return values

None	

5.1.2.8.2.2 SystemInit()

```
void SystemInit (
             void  )
```

Setup the microcontroller system.

Return values

None	

Chapter 6

Class Documentation

6.1 lstate_t Struct Reference

Public Attributes

- light_t **t_north**
- light_t **t_south**
- light_t **t_west**
- light_t **t_east**
- light_t **p_north**
- light_t **p_west**

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

- Inc/lights_functions.h

Chapter 7

File Documentation

7.1 FreeRTOSConfig.h

```
00001 /* USER CODE BEGIN Header */
00002 /*
00003  * FreeRTOS Kernel V10.3.1
00004  * Portion Copyright (C) 2017 Amazon.com, Inc. or its affiliates.  All Rights Reserved.
00005  * Portion Copyright (C) 2019 StMicroelectronics, Inc.  All Rights Reserved.
00006  *
00007  * Permission is hereby granted, free of charge, to any person obtaining a copy of
00008  * this software and associated documentation files (the "Software"), to deal in
00009  * the Software without restriction, including without limitation the rights to
00010  * use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
00011  * the Software, and to permit persons to whom the Software is furnished to do so,
00012  * subject to the following conditions:
00013  *
00014  * The above copyright notice and this permission notice shall be included in all
00015  * copies or substantial portions of the Software.
00016  *
00017  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00018  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
00019  * FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
00020  * COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
00021  * IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
00022  * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00023  *
00024  * http://www.FreeRTOS.org
00025  * http://aws.amazon.com/freertos
00026  *
00027  * 1 tab == 4 spaces!
00028  */
00029 /* USER CODE END Header */
00030
00031 #ifndef FREERTOS_CONFIG_H
00032 #define FREERTOS_CONFIG_H
00033
00034 /*-----------------------------------------------------------
00035  * Application specific definitions.
00036  *
00037  * These definitions should be adjusted for your particular hardware and
00038  * application requirements.
00039  *
00040  * These parameters and more are described within the 'configuration' section of the
00041  * FreeRTOS API documentation available on the FreeRTOS.org web site.
00042  *
00043  * See http://www.freertos.org/a00110.html
00044  *----------------------------------------------------------*/
00045
00046 /* USER CODE BEGIN Includes */
00047 /* Section where include file can be added */
00048 /* USER CODE END Includes */
00049
00050 /* Ensure definitions are only used by the compiler, and not by the assembler. */
00051 #if defined(__ICCARM__) || defined(__CC_ARM) || defined(__GNUC__)
00052   #include <stdint.h>
00053   extern uint32_t SystemCoreClock;
00054 #endif
00055 #ifndef CMSIS_device_header
00056 #define CMSIS_device_header "stm32l4xx.h"
00057 #endif /* CMSIS_device_header */
00058
```

```
00059 #define configENABLE_FPU                         0
00060 #define configENABLE_MPU                         0
00061
00062 #define configUSE_PREEMPTION                     1
00063 #define configSUPPORT_STATIC_ALLOCATION          1
00064 #define configSUPPORT_DYNAMIC_ALLOCATION         1
00065 #define configUSE_IDLE_HOOK                      0
00066 #define configUSE_TICK_HOOK                      0
00067 #define configCPU_CLOCK_HZ                       ( SystemCoreClock )
00068 #define configTICK_RATE_HZ                       ((TickType_t)1000)
00069 #define configMAX_PRIORITIES                     ( 56 )
00070 #define configMINIMAL_STACK_SIZE                 ((uint16_t)128)
00071 #define configTOTAL_HEAP_SIZE                    ((size_t)8000)
00072 #define configMAX_TASK_NAME_LEN                  ( 16 )
00073 #define configUSE_TRACE_FACILITY                 1
00074 #define configUSE_16_BIT_TICKS                   0
00075 #define configUSE_MUTEXES                        1
00076 #define configQUEUE_REGISTRY_SIZE                8
00077 #define configUSE_RECURSIVE_MUTEXES              1
00078 #define configUSE_COUNTING_SEMAPHORES            1
00079 #define configUSE_PORT_OPTIMISED_TASK_SELECTION  0
00080 /* USER CODE BEGIN MESSAGE_BUFFER_LENGTH_TYPE */
00081 /* Defaults to size_t for backward compatibility, but can be changed
00082    if lengths will always be less than the number of bytes in a size_t. */
00083 #define configMESSAGE_BUFFER_LENGTH_TYPE         size_t
00084 /* USER CODE END MESSAGE_BUFFER_LENGTH_TYPE */
00085
00086 /* Co-routine definitions. */
00087 #define configUSE_CO_ROUTINES                    0
00088 #define configMAX_CO_ROUTINE_PRIORITIES          ( 2 )
00089
00090 /* Software timer definitions. */
00091 #define configUSE_TIMERS                         1
00092 #define configTIMER_TASK_PRIORITY                ( 2 )
00093 #define configTIMER_QUEUE_LENGTH                 10
00094 #define configTIMER_TASK_STACK_DEPTH             256
00095
00096 /* The following flag must be enabled only when using newlib */
00097 #define configUSE_NEWLIB_REENTRANT          1
00098
00099 /* CMSIS-RTOS V2 flags */
00100 #define configUSE_OS2_THREAD_SUSPEND_RESUME  1
00101 #define configUSE_OS2_THREAD_ENUMERATE       1
00102 #define configUSE_OS2_EVENTFLAGS_FROM_ISR    1
00103 #define configUSE_OS2_THREAD_FLAGS           1
00104 #define configUSE_OS2_TIMER                  1
00105 #define configUSE_OS2_MUTEX                  1
00106
00107 /* Set the following definitions to 1 to include the API function, or zero
00108 to exclude the API function. */
00109 #define INCLUDE_vTaskPrioritySet             1
00110 #define INCLUDE_uxTaskPriorityGet            1
00111 #define INCLUDE_vTaskDelete                  1
00112 #define INCLUDE_vTaskCleanUpResources        0
00113 #define INCLUDE_vTaskSuspend                 1
00114 #define INCLUDE_vTaskDelayUntil              1
00115 #define INCLUDE_vTaskDelay                   1
00116 #define INCLUDE_xTaskGetSchedulerState       1
00117 #define INCLUDE_xTimerPendFunctionCall       1
00118 #define INCLUDE_xQueueGetMutexHolder         1
00119 #define INCLUDE_uxTaskGetStackHighWaterMark  1
00120 #define INCLUDE_xTaskGetCurrentTaskHandle    1
00121 #define INCLUDE_eTaskGetState                1
00122
00123 /*
00124  * The CMSIS-RTOS V2 FreeRTOS wrapper is dependent on the heap implementation used
00125  * by the application thus the correct define need to be enabled below
00126  */
00127 #define USE_FreeRTOS_HEAP_4
00128
00129 /* Cortex-M specific definitions. */
00130 #ifdef __NVIC_PRIO_BITS
00131  /* __BVIC_PRIO_BITS will be specified when CMSIS is being used. */
00132  #define configPRIO_BITS         __NVIC_PRIO_BITS
00133 #else
00134  #define configPRIO_BITS         4
00135 #endif
00136
00137 /* The lowest interrupt priority that can be used in a call to a "set priority"
00138 function. */
00139 #define configLIBRARY_LOWEST_INTERRUPT_PRIORITY   15
00140
00141 /* The highest interrupt priority that can be used by any interrupt service
00142 routine that makes calls to interrupt safe FreeRTOS API functions.  DO NOT CALL
00143 INTERRUPT SAFE FREERTOS API FUNCTIONS FROM ANY INTERRUPT THAT HAS A HIGHER
00144 PRIORITY THAN THIS! (higher priorities are lower numeric values. */
00145 #define configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY 5
```

```
00146
00147 /* Interrupt priorities used by the kernel port layer itself.  These are generic
00148 to all Cortex-M ports, and do not rely on any particular library functions. */
00149 #define configKERNEL_INTERRUPT_PRIORITY         ( configLIBRARY_LOWEST_INTERRUPT_PRIORITY << (8 -
      configPRIO_BITS) )
00150 /* !!!! configMAX_SYSCALL_INTERRUPT_PRIORITY must not be set to zero !!!!
00151 See http://www.FreeRTOS.org/RTOS-Cortex-M3-M4.html. */
00152 #define configMAX_SYSCALL_INTERRUPT_PRIORITY    ( configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY << (8 -
      configPRIO_BITS) )
00153
00154 /* Normal assert() semantics without relying on the provision of an assert.h
00155 header file. */
00156 /* USER CODE BEGIN 1 */
00157 #define configASSERT( x ) if ((x) == 0) {taskDISABLE_INTERRUPTS(); for( ;; );}
00158 /* USER CODE END 1 */
00159
00160 /* Definitions that map the FreeRTOS port interrupt handlers to their CMSIS
00161 standard names. */
00162 #define vPortSVCHandler    SVC_Handler
00163 #define xPortPendSVHandler PendSV_Handler
00164
00165 /* IMPORTANT: After 10.3.1 update, Systick_Handler comes from NVIC (if SYS timebase = systick),
      otherwise from cmsis_os2.c */
00166
00167 #define USE_CUSTOM_SYSTICK_HANDLER_IMPLEMENTATION 0
00168
00169 /* USER CODE BEGIN Defines */
00170 /* Section where parameter definitions can be added (for instance, to override default ones in
      FreeRTOS.h) */
00171 /* USER CODE END Defines */
00172
00173 #endif /* FREERTOS_CONFIG_H */
```

7.2 Inc/gpio.h File Reference

This file contains all the function prototypes for the gpio.c file.

```
#include "main.h"
```

Functions

- void MX_GPIO_Init (void)

7.2.1 Detailed Description

This file contains all the function prototypes for the gpio.c file.

Attention

7.2.2 Function Documentation

7.2.2.1 MX_GPIO_Init()

```
void MX_GPIO_Init (
             void  )
```

Configure pins as Analog Input Output EVENT_OUT EXTI

7.3 gpio.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00019 /* USER CODE END Header */
00020 /* Define to prevent recursive inclusion -------------------------------------*/
00021 #ifndef __GPIO_H__
00022 #define __GPIO_H__
00023
00024 #ifdef __cplusplus
00025 extern "C" {
00026 #endif
00027
00028 /* Includes ------------------------------------------------------------------*/
00029 #include "main.h"
00030
00031 /* USER CODE BEGIN Includes */
00032
00033 /* USER CODE END Includes */
00034
00035 /* USER CODE BEGIN Private defines */
00036
00037 /* USER CODE END Private defines */
00038
00039 void MX_GPIO_Init(void);
00040
00041 /* USER CODE BEGIN Prototypes */
00042
00043 /* USER CODE END Prototypes */
00044
00045 #ifdef __cplusplus
00046 }
00047 #endif
00048 #endif /*__ GPIO_H__ */
00049
```

7.4 lights_functions.h

```
00001
00002
00003 #pragma once
00004 #include "FreeRTOS.h"
00005 #include "cmsis_os.h"
00006 #include "gpio.h"
00007 #include "main.h"
00008 #include "spi.h"
00009 #include "usart.h"
00010
00011 #include <stdint.h>
00012
00013 #define SHIFT_REG_CAPACITY 3
00014 #define BTN_P_NORTH 0b00000001
00015 #define BTN_P_WEST 0b00000010
00016 #define BTN_T_NORTH 0b00000100
00017 #define BTN_T_EAST 0b00001000
00018 #define BTN_T_SOUTH 0b00010000
00019 #define BTN_T_WEST 0b00100000
00020
00021 typedef enum { red, green, yellow, blue, off } light_t;
00022
00023 typedef enum {north_south, east_west, pnorth, pwest, none, ok} direction_t;
00024
00025 typedef struct lstate_t {
```

```
00026   light_t t_north;
00027   light_t t_south;
00028   light_t t_west;
00029   light_t t_east;
00030   light_t p_north;
00031   light_t p_west;
00032 } lstate_t;
00033
00034
00035 void feed_shift_reg(uint8_t *lights);
00036 void format_lights(const lstate_t *curr, uint8_t *lights);
00037 uint8_t get_buttons();
00038 void toggle_blue(light_t *p_light);
00039 //void pedestrian_go(light_t *p_light);
00040 //void car_go(light_t *t_light);
```

7.5 lights_test.h

```
00001
00002 #pragma once
00003 #include "lights_functions.h"
00004
00005 void test_program(lstate_t *state);
00006 void test_feed_shift_ref();
00007 void test_format_lights();
00008 void test_get_buttons(lstate_t *state);
00009 void test_toggle_blue(lstate_t *state);
```

7.6 Inc/main.h File Reference

: Header for main.c file. This file contains the common defines of the application.

```
#include "stm32l4xx_hal.h"
```

Macros

- #define **B1_Pin** GPIO_PIN_13
- #define **B1_GPIO_Port** GPIOC
- #define **USART_TX_Pin** GPIO_PIN_2
- #define **USART_TX_GPIO_Port** GPIOA
- #define **USART_RX_Pin** GPIO_PIN_3
- #define **USART_RX_GPIO_Port** GPIOA
- #define **LD2_Pin** GPIO_PIN_5
- #define **LD2_GPIO_Port** GPIOA
- #define **T_West_Pin** GPIO_PIN_4
- #define **T_West_GPIO_Port** GPIOC
- #define **STCP_Pin** GPIO_PIN_12
- #define **STCP_GPIO_Port** GPIOB
- #define **T_South_Pin** GPIO_PIN_13
- #define **T_South_GPIO_Port** GPIOB
- #define **T_East_Pin** GPIO_PIN_14
- #define **T_East_GPIO_Port** GPIOB
- #define **Enable_Pin** GPIO_PIN_7
- #define **Enable_GPIO_Port** GPIOC
- #define **Reset_Pin** GPIO_PIN_9
- #define **Reset_GPIO_Port** GPIOA
- #define **T_North_Pin** GPIO_PIN_10
- #define **T_North_GPIO_Port** GPIOA

- #define **TMS_Pin** GPIO_PIN_13
- #define **TMS_GPIO_Port** GPIOA
- #define **TCK_Pin** GPIO_PIN_14
- #define **TCK_GPIO_Port** GPIOA
- #define **P_West_Pin** GPIO_PIN_15
- #define **P_West_GPIO_Port** GPIOA
- #define **SWO_Pin** GPIO_PIN_3
- #define **SWO_GPIO_Port** GPIOB
- #define **P_North_Pin** GPIO_PIN_7
- #define **P_North_GPIO_Port** GPIOB

Functions

- void Error_Handler (void)

  This function is executed in case of error occurrence.

7.6.1 Detailed Description

: Header for main.c file. This file contains the common defines of the application.

Attention

7.6.2 Function Documentation

7.6.2.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

7.7 main.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
```

```
00019 /* USER CODE END Header */
00020
00021 /* Define to prevent recursive inclusion -------------------------------------*/
00022 #ifndef __MAIN_H
00023 #define __MAIN_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif
00028
00029 /* Includes ------------------------------------------------------------------*/
00030 #include "stm32l4xx_hal.h"
00031
00032 /* Private includes ----------------------------------------------------------*/
00033 /* USER CODE BEGIN Includes */
00034
00035 /* USER CODE END Includes */
00036
00037 /* Exported types ------------------------------------------------------------*/
00038 /* USER CODE BEGIN ET */
00039
00040 /* USER CODE END ET */
00041
00042 /* Exported constants --------------------------------------------------------*/
00043 /* USER CODE BEGIN EC */
00044
00045 /* USER CODE END EC */
00046
00047 /* Exported macro ------------------------------------------------------------*/
00048 /* USER CODE BEGIN EM */
00049
00050 /* USER CODE END EM */
00051
00052 /* Exported functions prototypes ---------------------------------------------*/
00053 void Error_Handler(void);
00054
00055 /* USER CODE BEGIN EFP */
00056
00057 /* USER CODE END EFP */
00058
00059 /* Private defines -----------------------------------------------------------*/
00060 #define B1_Pin GPIO_PIN_13
00061 #define B1_GPIO_Port GPIOC
00062 #define USART_TX_Pin GPIO_PIN_2
00063 #define USART_TX_GPIO_Port GPIOA
00064 #define USART_RX_Pin GPIO_PIN_3
00065 #define USART_RX_GPIO_Port GPIOA
00066 #define LD2_Pin GPIO_PIN_5
00067 #define LD2_GPIO_Port GPIOA
00068 #define T_West_Pin GPIO_PIN_4
00069 #define T_West_GPIO_Port GPIOC
00070 #define STCP_Pin GPIO_PIN_12
00071 #define STCP_GPIO_Port GPIOB
00072 #define T_South_Pin GPIO_PIN_13
00073 #define T_South_GPIO_Port GPIOB
00074 #define T_East_Pin GPIO_PIN_14
00075 #define T_East_GPIO_Port GPIOB
00076 #define Enable_Pin GPIO_PIN_7
00077 #define Enable_GPIO_Port GPIOC
00078 #define Reset_Pin GPIO_PIN_9
00079 #define Reset_GPIO_Port GPIOA
00080 #define T_North_Pin GPIO_PIN_10
00081 #define T_North_GPIO_Port GPIOA
00082 #define TMS_Pin GPIO_PIN_13
00083 #define TMS_GPIO_Port GPIOA
00084 #define TCK_Pin GPIO_PIN_14
00085 #define TCK_GPIO_Port GPIOA
00086 #define P_West_Pin GPIO_PIN_15
00087 #define P_West_GPIO_Port GPIOA
00088 #define SWO_Pin GPIO_PIN_3
00089 #define SWO_GPIO_Port GPIOB
00090 #define P_North_Pin GPIO_PIN_7
00091 #define P_North_GPIO_Port GPIOB
00092
00093 /* USER CODE BEGIN Private defines */
00094
00095 /* USER CODE END Private defines */
00096
00097 #ifdef __cplusplus
00098 }
00099 #endif
00100
00101 #endif /* __MAIN_H */
```

7.8 Inc/spi.h File Reference

This file contains all the function prototypes for the spi.c file.

```
#include "main.h"
```

Functions

- void **MX_SPI3_Init** (void)

Variables

- SPI_HandleTypeDef **hspi3**

7.8.1 Detailed Description

This file contains all the function prototypes for the spi.c file.

Attention

7.9 spi.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00019 /* USER CODE END Header */
00020 /* Define to prevent recursive inclusion -------------------------------------*/
00021 #ifndef __SPI_H__
00022 #define __SPI_H__
00023
00024 #ifdef __cplusplus
00025 extern "C" {
00026 #endif
00027
00028 /* Includes ------------------------------------------------------------------*/
00029 #include "main.h"
00030
00031 /* USER CODE BEGIN Includes */
00032
00033 /* USER CODE END Includes */
00034
00035 extern SPI_HandleTypeDef hspi3;
00036
00037 /* USER CODE BEGIN Private defines */
00038
00039 /* USER CODE END Private defines */
00040
00041 void MX_SPI3_Init(void);
00042
00043 /* USER CODE BEGIN Prototypes */
00044
00045 /* USER CODE END Prototypes */
00046
00047 #ifdef __cplusplus
00048 }
00049 #endif
00050
00051 #endif /* __SPI_H__ */
00052
```

7.10 Inc/stm32l4xx_hal_conf.h File Reference

HAL configuration template file. This file should be copied to the application folder and renamed to stm32l4xx_hal_conf.h.

```
#include "stm32l4xx_hal_rcc.h"
#include "stm32l4xx_hal_gpio.h"
#include "stm32l4xx_hal_dma.h"
#include "stm32l4xx_hal_cortex.h"
#include "stm32l4xx_hal_exti.h"
#include "stm32l4xx_hal_flash.h"
#include "stm32l4xx_hal_pwr.h"
#include "stm32l4xx_hal_spi.h"
#include "stm32l4xx_hal_tim.h"
#include "stm32l4xx_hal_uart.h"
```

Macros

- #define **HAL_MODULE_ENABLED**

  This is the list of modules to be used in the HAL driver.
- #define **HAL_SPI_MODULE_ENABLED**
- #define **HAL_TIM_MODULE_ENABLED**
- #define **HAL_UART_MODULE_ENABLED**
- #define **HAL_GPIO_MODULE_ENABLED**
- #define **HAL_EXTI_MODULE_ENABLED**
- #define **HAL_DMA_MODULE_ENABLED**
- #define **HAL_RCC_MODULE_ENABLED**
- #define **HAL_FLASH_MODULE_ENABLED**
- #define **HAL_PWR_MODULE_ENABLED**
- #define **HAL_CORTEX_MODULE_ENABLED**
- #define HSE_VALUE ((uint32_t)8000000U)

  Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).
- #define HSE_STARTUP_TIMEOUT ((uint32_t)100U)
- #define MSI_VALUE ((uint32_t)4000000U)

  Internal Multiple Speed oscillator (MSI) default value. This value is the default MSI range value after Reset.
- #define HSI_VALUE ((uint32_t)16000000U)

  Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).
- #define HSI48_VALUE ((uint32_t)48000000U)

  Internal High Speed oscillator (HSI48) value for USB FS, SDMMC and RNG. This internal oscillator is mainly dedicated to provide a high precision clock to the USB peripheral by means of a special Clock Recovery System (CRS) circuitry. When the CRS is not used, the HSI48 RC oscillator runs on it default frequency which is subject to manufacturing process variations.
- #define LSI_VALUE 32000U

  Internal Low Speed oscillator (LSI) value.
- #define LSE_VALUE 32768U

  External Low Speed oscillator (LSE) value. This value is used by the UART, RTC HAL module to compute the system frequency.
- #define LSE_STARTUP_TIMEOUT 5000U
- #define EXTERNAL_SAI1_CLOCK_VALUE 2097000U

  External clock source for SAI1 peripheral This value is used by the RCC HAL module to compute the SAI1 & SAI2 clock source frequency.

- #define EXTERNAL_SAI2_CLOCK_VALUE 2097000U

  External clock source for SAI2 peripheral This value is used by the RCC HAL module to compute the SAI1 & SAI2 clock source frequency.
- #define VDD_VALUE 3300U

  This is the HAL system configuration section.
- #define TICK_INT_PRIORITY 15U
- #define **USE_RTOS** 0U
- #define **PREFETCH_ENABLE** 1U
- #define **INSTRUCTION_CACHE_ENABLE** 1U
- #define **DATA_CACHE_ENABLE** 1U
- #define USE_HAL_ADC_REGISTER_CALLBACKS 0U

  Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.
- #define **USE_HAL_CAN_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_COMP_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_CRYP_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_DAC_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_DCMI_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_DFSDM_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_DMA2D_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_DSI_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_GFXMMU_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_HASH_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_HCD_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_I2C_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_IRDA_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_LPTIM_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_LTDC_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_MMC_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_OPAMP_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_OSPI_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_PCD_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_QSPI_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_RNG_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_RTC_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_SAI_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_SD_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_SMARTCARD_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_SMBUS_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_SPI_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_SWPMI_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_TIM_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_TSC_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_UART_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_USART_REGISTER_CALLBACKS** 0U
- #define **USE_HAL_WWDG_REGISTER_CALLBACKS** 0U
- #define **USE_SPI_CRC** 0U
- #define **assert_param**(expr) ((void)0U)

  Include module's header file.

7.10.1 Detailed Description

HAL configuration template file. This file should be copied to the application folder and renamed to stm32l4xx_hal_conf.h.

Author

MCD Application Team

Attention

7.10.2 Macro Definition Documentation

7.10.2.1 EXTERNAL_SAI1_CLOCK_VALUE

```
#define EXTERNAL_SAI1_CLOCK_VALUE 2097000U
```

External clock source for SAI1 peripheral This value is used by the RCC HAL module to compute the SAI1 & SAI2 clock source frequency.

Value of the SAI1 External clock source in Hz

7.10.2.2 EXTERNAL_SAI2_CLOCK_VALUE

```
#define EXTERNAL_SAI2_CLOCK_VALUE 2097000U
```

External clock source for SAI2 peripheral This value is used by the RCC HAL module to compute the SAI1 & SAI2 clock source frequency.

Value of the SAI2 External clock source in Hz

7.10.2.3 HSE_STARTUP_TIMEOUT

```
#define HSE_STARTUP_TIMEOUT ((uint32_t)100U)
```

Time out for HSE start up, in ms

7.10.2.4 HSE_VALUE

```
#define HSE_VALUE ((uint32_t)8000000U)
```

Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).

Value of the External oscillator in Hz

7.10.2.5 HSI48_VALUE

```
#define HSI48_VALUE ((uint32_t)48000000U)
```

Internal High Speed oscillator (HSI48) value for USB FS, SDMMC and RNG. This internal oscillator is mainly dedicated to provide a high precision clock to the USB peripheral by means of a special Clock Recovery System (CRS) circuitry. When the CRS is not used, the HSI48 RC oscillator runs on it default frequency which is subject to manufacturing process variations.

Value of the Internal High Speed oscillator for USB FS/SDMMC/RNG in Hz. The real value my vary depending on manufacturing process variations.

7.10.2.6 HSI_VALUE

```
#define HSI_VALUE ((uint32_t)16000000U)
```

Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).

Value of the Internal oscillator in Hz

7.10.2.7 LSE_STARTUP_TIMEOUT

```
#define LSE_STARTUP_TIMEOUT 5000U
```

Time out for LSE start up, in ms

7.10.2.8 LSE_VALUE

```
#define LSE_VALUE 32768U
```

External Low Speed oscillator (LSE) value. This value is used by the UART, RTC HAL module to compute the system frequency.

$<$ Value of the Internal Low Speed oscillator in Hz The real value may vary depending on the variations in voltage and temperature. Value of the External oscillator in Hz

7.10.2.9 LSI_VALUE

```
#define LSI_VALUE 32000U
```

Internal Low Speed oscillator (LSI) value.

LSI Typical Value in Hz

7.10.2.10 MSI_VALUE

```
#define MSI_VALUE ((uint32_t)4000000U)
```

Internal Multiple Speed oscillator (MSI) default value. This value is the default MSI range value after Reset.

Value of the Internal oscillator in Hz

7.10.2.11 TICK_INT_PRIORITY

```
#define TICK_INT_PRIORITY 15U
```

tick interrupt priority

7.10.2.12 USE_HAL_ADC_REGISTER_CALLBACKS

```
#define USE_HAL_ADC_REGISTER_CALLBACKS 0U
```

Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.

Set below the peripheral configuration to "1U" to add the support of HAL callback registration/deregistration feature for the HAL driver(s). This allows user application to provide specific callback functions thanks to HAL_PPP_↩ RegisterCallback() rather than overwriting the default weak callback functions (see each stm32l4xx_hal_ppp.h file for possible callback identifiers defined in HAL_PPP_CallbackIDTypeDef for each PPP peripheral).

7.10.2.13 VDD_VALUE

```
#define VDD_VALUE 3300U
```

This is the HAL system configuration section.

Value of VDD in mv

7.11 stm32l4xx_hal_conf.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00021 /* USER CODE END Header */
00022
00023 /* Define to prevent recursive inclusion -------------------------------------*/
00024 #ifndef STM32L4xx_HAL_CONF_H
00025 #define STM32L4xx_HAL_CONF_H
00026
00027 #ifdef __cplusplus
00028  extern "C" {
00029 #endif
00030
00031 /* Exported types ------------------------------------------------------------*/
00032 /* Exported constants --------------------------------------------------------*/
00033
00034 /* ########################## Module Selection ############################## */
00038 #define HAL_MODULE_ENABLED
00039 /*#define HAL_ADC_MODULE_ENABLED   */
00040 /*#define HAL_CRYP_MODULE_ENABLED   */
00041 /*#define HAL_CAN_MODULE_ENABLED   */
00042 /*#define HAL_COMP_MODULE_ENABLED   */
00043 /*#define HAL_I2C_MODULE_ENABLED   */
00044 /*#define HAL_CRC_MODULE_ENABLED   */
00045 /*#define HAL_CRYP_MODULE_ENABLED   */
00046 /*#define HAL_DAC_MODULE_ENABLED   */
00047 /*#define HAL_DCMI_MODULE_ENABLED   */
00048 /*#define HAL_DMA2D_MODULE_ENABLED   */
00049 /*#define HAL_DFSDM_MODULE_ENABLED   */
00050 /*#define HAL_DSI_MODULE_ENABLED   */
00051 /*#define HAL_FIREWALL_MODULE_ENABLED   */
00052 /*#define HAL_GFXMMU_MODULE_ENABLED   */
00053 /*#define HAL_HCD_MODULE_ENABLED   */
00054 /*#define HAL_HASH_MODULE_ENABLED   */
00055 /*#define HAL_I2S_MODULE_ENABLED   */
00056 /*#define HAL_IRDA_MODULE_ENABLED   */
00057 /*#define HAL_IWDG_MODULE_ENABLED   */
00058 /*#define HAL_LTDC_MODULE_ENABLED   */
```

```
00059 /*#define HAL_LCD_MODULE_ENABLED   */
00060 /*#define HAL_LPTIM_MODULE_ENABLED   */
00061 /*#define HAL_MMC_MODULE_ENABLED   */
00062 /*#define HAL_NAND_MODULE_ENABLED   */
00063 /*#define HAL_NOR_MODULE_ENABLED   */
00064 /*#define HAL_OPAMP_MODULE_ENABLED   */
00065 /*#define HAL_OSPI_MODULE_ENABLED   */
00066 /*#define HAL_OSPI_MODULE_ENABLED   */
00067 /*#define HAL_PCD_MODULE_ENABLED   */
00068 /*#define HAL_PKA_MODULE_ENABLED   */
00069 /*#define HAL_QSPI_MODULE_ENABLED   */
00070 /*#define HAL_QSPI_MODULE_ENABLED   */
00071 /*#define HAL_RNG_MODULE_ENABLED   */
00072 /*#define HAL_RTC_MODULE_ENABLED   */
00073 /*#define HAL_SAI_MODULE_ENABLED   */
00074 /*#define HAL_SD_MODULE_ENABLED   */
00075 /*#define HAL_SMBUS_MODULE_ENABLED   */
00076 /*#define HAL_SMARTCARD_MODULE_ENABLED   */
00077 #define HAL_SPI_MODULE_ENABLED
00078 /*#define HAL_SRAM_MODULE_ENABLED   */
00079 /*#define HAL_SWPMI_MODULE_ENABLED   */
00080 #define HAL_TIM_MODULE_ENABLED
00081 /*#define HAL_TSC_MODULE_ENABLED   */
00082 #define HAL_UART_MODULE_ENABLED
00083 /*#define HAL_USART_MODULE_ENABLED   */
00084 /*#define HAL_WWDG_MODULE_ENABLED   */
00085 /*#define HAL_EXTI_MODULE_ENABLED   */
00086 /*#define HAL_PSSI_MODULE_ENABLED   */
00087 #define HAL_GPIO_MODULE_ENABLED
00088 #define HAL_EXTI_MODULE_ENABLED
00089 #define HAL_DMA_MODULE_ENABLED
00090 #define HAL_RCC_MODULE_ENABLED
00091 #define HAL_FLASH_MODULE_ENABLED
00092 #define HAL_PWR_MODULE_ENABLED
00093 #define HAL_CORTEX_MODULE_ENABLED
00094
00095 /* ########################## Oscillator Values adaptation ####################*/
00101 #if !defined  (HSE_VALUE)
00102   #define HSE_VALUE    ((uint32_t)8000000U)
00103 #endif /* HSE_VALUE */
00104
00105 #if !defined  (HSE_STARTUP_TIMEOUT)
00106   #define HSE_STARTUP_TIMEOUT    ((uint32_t)100U)
00107 #endif /* HSE_STARTUP_TIMEOUT */
00108
00113 #if !defined  (MSI_VALUE)
00114   #define MSI_VALUE    ((uint32_t)4000000U)
00115 #endif /* MSI_VALUE */
00121 #if !defined  (HSI_VALUE)
00122   #define HSI_VALUE    ((uint32_t)16000000U)
00123 #endif /* HSI_VALUE */
00124
00132 #if !defined  (HSI48_VALUE)
00133  #define HSI48_VALUE   ((uint32_t)48000000U)
00135 #endif /* HSI48_VALUE */
00136
00140 #if !defined  (LSI_VALUE)
00141  #define LSI_VALUE  32000U
00142 #endif /* LSI_VALUE */
00150 #if !defined  (LSE_VALUE)
00151   #define LSE_VALUE    32768U
00152 #endif /* LSE_VALUE */
00153
00154 #if !defined  (LSE_STARTUP_TIMEOUT)
00155   #define LSE_STARTUP_TIMEOUT    5000U
00156 #endif /* HSE_STARTUP_TIMEOUT */
00157
00163 #if !defined  (EXTERNAL_SAI1_CLOCK_VALUE)
00164   #define EXTERNAL_SAI1_CLOCK_VALUE    2097000U
00165 #endif /* EXTERNAL_SAI1_CLOCK_VALUE */
00166
00172 #if !defined  (EXTERNAL_SAI2_CLOCK_VALUE)
00173   #define EXTERNAL_SAI2_CLOCK_VALUE    2097000U
00174 #endif /* EXTERNAL_SAI2_CLOCK_VALUE */
00175
00176 /* Tip: To avoid modifying this file each time you need to use different HSE,
00177    ===  you can define the HSE value in your toolchain compiler preprocessor. */
00178
00179 /* ########################## System Configuration ######################### */
00184 #define  VDD_VALUE                    3300U
00185 #define  TICK_INT_PRIORITY            15U
00186 #define  USE_RTOS                     0U
00187 #define  PREFETCH_ENABLE              1U
00188 #define  INSTRUCTION_CACHE_ENABLE     1U
00189 #define  DATA_CACHE_ENABLE            1U
00190
00191 /* ########################## Assert Selection ############################## */
```

```
00196 /* #define USE_FULL_ASSERT    1U */
00197
00198 /* ################### Register callback feature configuration ############### */
00208 #define USE_HAL_ADC_REGISTER_CALLBACKS         0U
00209 #define USE_HAL_CAN_REGISTER_CALLBACKS         0U
00210 #define USE_HAL_COMP_REGISTER_CALLBACKS        0U
00211 #define USE_HAL_CRYP_REGISTER_CALLBACKS        0U
00212 #define USE_HAL_DAC_REGISTER_CALLBACKS         0U
00213 #define USE_HAL_DCMI_REGISTER_CALLBACKS        0U
00214 #define USE_HAL_DFSDM_REGISTER_CALLBACKS       0U
00215 #define USE_HAL_DMA2D_REGISTER_CALLBACKS       0U
00216 #define USE_HAL_DSI_REGISTER_CALLBACKS         0U
00217 #define USE_HAL_GFXMMU_REGISTER_CALLBACKS      0U
00218 #define USE_HAL_HASH_REGISTER_CALLBACKS        0U
00219 #define USE_HAL_HCD_REGISTER_CALLBACKS         0U
00220 #define USE_HAL_I2C_REGISTER_CALLBACKS         0U
00221 #define USE_HAL_IRDA_REGISTER_CALLBACKS        0U
00222 #define USE_HAL_LPTIM_REGISTER_CALLBACKS       0U
00223 #define USE_HAL_LTDC_REGISTER_CALLBACKS        0U
00224 #define USE_HAL_MMC_REGISTER_CALLBACKS         0U
00225 #define USE_HAL_OPAMP_REGISTER_CALLBACKS       0U
00226 #define USE_HAL_OSPI_REGISTER_CALLBACKS        0U
00227 #define USE_HAL_PCD_REGISTER_CALLBACKS         0U
00228 #define USE_HAL_QSPI_REGISTER_CALLBACKS        0U
00229 #define USE_HAL_RNG_REGISTER_CALLBACKS         0U
00230 #define USE_HAL_RTC_REGISTER_CALLBACKS         0U
00231 #define USE_HAL_SAI_REGISTER_CALLBACKS         0U
00232 #define USE_HAL_SD_REGISTER_CALLBACKS          0U
00233 #define USE_HAL_SMARTCARD_REGISTER_CALLBACKS   0U
00234 #define USE_HAL_SMBUS_REGISTER_CALLBACKS       0U
00235 #define USE_HAL_SPI_REGISTER_CALLBACKS         0U
00236 #define USE_HAL_SWPMI_REGISTER_CALLBACKS       0U
00237 #define USE_HAL_TIM_REGISTER_CALLBACKS         0U
00238 #define USE_HAL_TSC_REGISTER_CALLBACKS         0U
00239 #define USE_HAL_UART_REGISTER_CALLBACKS        0U
00240 #define USE_HAL_USART_REGISTER_CALLBACKS       0U
00241 #define USE_HAL_WWDG_REGISTER_CALLBACKS        0U
00242
00243 /* ################## SPI peripheral configuration ########################### */
00244
00245 /* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver
00246  * Activated: CRC code is present inside driver
00247  * Deactivated: CRC code cleaned from driver
00248  */
00249
00250 #define USE_SPI_CRC                    0U
00251
00252 /* Includes ------------------------------------------------------------------*/
00257 #ifdef HAL_RCC_MODULE_ENABLED
00258   #include "stm32l4xx_hal_rcc.h"
00259 #endif /* HAL_RCC_MODULE_ENABLED */
00260
00261 #ifdef HAL_GPIO_MODULE_ENABLED
00262   #include "stm32l4xx_hal_gpio.h"
00263 #endif /* HAL_GPIO_MODULE_ENABLED */
00264
00265 #ifdef HAL_DMA_MODULE_ENABLED
00266   #include "stm32l4xx_hal_dma.h"
00267 #endif /* HAL_DMA_MODULE_ENABLED */
00268
00269 #ifdef HAL_DFSDM_MODULE_ENABLED
00270   #include "stm32l4xx_hal_dfsdm.h"
00271 #endif /* HAL_DFSDM_MODULE_ENABLED */
00272
00273 #ifdef HAL_CORTEX_MODULE_ENABLED
00274   #include "stm32l4xx_hal_cortex.h"
00275 #endif /* HAL_CORTEX_MODULE_ENABLED */
00276
00277 #ifdef HAL_ADC_MODULE_ENABLED
00278   #include "stm32l4xx_hal_adc.h"
00279 #endif /* HAL_ADC_MODULE_ENABLED */
00280
00281 #ifdef HAL_CAN_MODULE_ENABLED
00282   #include "stm32l4xx_hal_can.h"
00283 #endif /* HAL_CAN_MODULE_ENABLED */
00284
00285 #ifdef HAL_CAN_LEGACY_MODULE_ENABLED
00286   #include "Legacy/stm32l4xx_hal_can_legacy.h"
00287 #endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
00288
00289 #ifdef HAL_COMP_MODULE_ENABLED
00290   #include "stm32l4xx_hal_comp.h"
00291 #endif /* HAL_COMP_MODULE_ENABLED */
00292
00293 #ifdef HAL_CRC_MODULE_ENABLED
00294   #include "stm32l4xx_hal_crc.h"
00295 #endif /* HAL_CRC_MODULE_ENABLED */
```

```
00296
00297 #ifdef HAL_CRYP_MODULE_ENABLED
00298   #include "stm32l4xx_hal_cryp.h"
00299 #endif /* HAL_CRYP_MODULE_ENABLED */
00300
00301 #ifdef HAL_DAC_MODULE_ENABLED
00302   #include "stm32l4xx_hal_dac.h"
00303 #endif /* HAL_DAC_MODULE_ENABLED */
00304
00305 #ifdef HAL_DCMI_MODULE_ENABLED
00306   #include "stm32l4xx_hal_dcmi.h"
00307 #endif /* HAL_DCMI_MODULE_ENABLED */
00308
00309 #ifdef HAL_DMA2D_MODULE_ENABLED
00310   #include "stm32l4xx_hal_dma2d.h"
00311 #endif /* HAL_DMA2D_MODULE_ENABLED */
00312
00313 #ifdef HAL_DSI_MODULE_ENABLED
00314   #include "stm32l4xx_hal_dsi.h"
00315 #endif /* HAL_DSI_MODULE_ENABLED */
00316
00317 #ifdef HAL_EXTI_MODULE_ENABLED
00318   #include "stm32l4xx_hal_exti.h"
00319 #endif /* HAL_EXTI_MODULE_ENABLED */
00320
00321 #ifdef HAL_GFXMMU_MODULE_ENABLED
00322   #include "stm32l4xx_hal_gfxmmu.h"
00323 #endif /* HAL_GFXMMU_MODULE_ENABLED */
00324
00325 #ifdef HAL_FIREWALL_MODULE_ENABLED
00326   #include "stm32l4xx_hal_firewall.h"
00327 #endif /* HAL_FIREWALL_MODULE_ENABLED */
00328
00329 #ifdef HAL_FLASH_MODULE_ENABLED
00330   #include "stm32l4xx_hal_flash.h"
00331 #endif /* HAL_FLASH_MODULE_ENABLED */
00332
00333 #ifdef HAL_HASH_MODULE_ENABLED
00334   #include "stm32l4xx_hal_hash.h"
00335 #endif /* HAL_HASH_MODULE_ENABLED */
00336
00337 #ifdef HAL_HCD_MODULE_ENABLED
00338   #include "stm32l4xx_hal_hcd.h"
00339 #endif /* HAL_HCD_MODULE_ENABLED */
00340
00341 #ifdef HAL_I2C_MODULE_ENABLED
00342   #include "stm32l4xx_hal_i2c.h"
00343 #endif /* HAL_I2C_MODULE_ENABLED */
00344
00345 #ifdef HAL_IRDA_MODULE_ENABLED
00346   #include "stm32l4xx_hal_irda.h"
00347 #endif /* HAL_IRDA_MODULE_ENABLED */
00348
00349 #ifdef HAL_IWDG_MODULE_ENABLED
00350   #include "stm32l4xx_hal_iwdg.h"
00351 #endif /* HAL_IWDG_MODULE_ENABLED */
00352
00353 #ifdef HAL_LCD_MODULE_ENABLED
00354   #include "stm32l4xx_hal_lcd.h"
00355 #endif /* HAL_LCD_MODULE_ENABLED */
00356
00357 #ifdef HAL_LPTIM_MODULE_ENABLED
00358   #include "stm32l4xx_hal_lptim.h"
00359 #endif /* HAL_LPTIM_MODULE_ENABLED */
00360
00361 #ifdef HAL_LTDC_MODULE_ENABLED
00362   #include "stm32l4xx_hal_ltdc.h"
00363 #endif /* HAL_LTDC_MODULE_ENABLED */
00364
00365 #ifdef HAL_MMC_MODULE_ENABLED
00366   #include "stm32l4xx_hal_mmc.h"
00367 #endif /* HAL_MMC_MODULE_ENABLED */
00368
00369 #ifdef HAL_NAND_MODULE_ENABLED
00370   #include "stm32l4xx_hal_nand.h"
00371 #endif /* HAL_NAND_MODULE_ENABLED */
00372
00373 #ifdef HAL_NOR_MODULE_ENABLED
00374   #include "stm32l4xx_hal_nor.h"
00375 #endif /* HAL_NOR_MODULE_ENABLED */
00376
00377 #ifdef HAL_OPAMP_MODULE_ENABLED
00378   #include "stm32l4xx_hal_opamp.h"
00379 #endif /* HAL_OPAMP_MODULE_ENABLED */
00380
00381 #ifdef HAL_OSPI_MODULE_ENABLED
00382   #include "stm32l4xx_hal_ospi.h"
```

```
00383 #endif /* HAL_OSPI_MODULE_ENABLED */
00384
00385 #ifdef HAL_PCD_MODULE_ENABLED
00386   #include "stm32l4xx_hal_pcd.h"
00387 #endif /* HAL_PCD_MODULE_ENABLED */
00388
00389 #ifdef HAL_PKA_MODULE_ENABLED
00390   #include "stm32l4xx_hal_pka.h"
00391 #endif /* HAL_PKA_MODULE_ENABLED */
00392
00393 #ifdef HAL_PSSI_MODULE_ENABLED
00394   #include "stm32l4xx_hal_pssi.h"
00395 #endif /* HAL_PSSI_MODULE_ENABLED */
00396
00397 #ifdef HAL_PWR_MODULE_ENABLED
00398   #include "stm32l4xx_hal_pwr.h"
00399 #endif /* HAL_PWR_MODULE_ENABLED */
00400
00401 #ifdef HAL_QSPI_MODULE_ENABLED
00402   #include "stm32l4xx_hal_qspi.h"
00403 #endif /* HAL_QSPI_MODULE_ENABLED */
00404
00405 #ifdef HAL_RNG_MODULE_ENABLED
00406   #include "stm32l4xx_hal_rng.h"
00407 #endif /* HAL_RNG_MODULE_ENABLED */
00408
00409 #ifdef HAL_RTC_MODULE_ENABLED
00410   #include "stm32l4xx_hal_rtc.h"
00411 #endif /* HAL_RTC_MODULE_ENABLED */
00412
00413 #ifdef HAL_SAI_MODULE_ENABLED
00414   #include "stm32l4xx_hal_sai.h"
00415 #endif /* HAL_SAI_MODULE_ENABLED */
00416
00417 #ifdef HAL_SD_MODULE_ENABLED
00418   #include "stm32l4xx_hal_sd.h"
00419 #endif /* HAL_SD_MODULE_ENABLED */
00420
00421 #ifdef HAL_SMARTCARD_MODULE_ENABLED
00422   #include "stm32l4xx_hal_smartcard.h"
00423 #endif /* HAL_SMARTCARD_MODULE_ENABLED */
00424
00425 #ifdef HAL_SMBUS_MODULE_ENABLED
00426   #include "stm32l4xx_hal_smbus.h"
00427 #endif /* HAL_SMBUS_MODULE_ENABLED */
00428
00429 #ifdef HAL_SPI_MODULE_ENABLED
00430   #include "stm32l4xx_hal_spi.h"
00431 #endif /* HAL_SPI_MODULE_ENABLED */
00432
00433 #ifdef HAL_SRAM_MODULE_ENABLED
00434   #include "stm32l4xx_hal_sram.h"
00435 #endif /* HAL_SRAM_MODULE_ENABLED */
00436
00437 #ifdef HAL_SWPMI_MODULE_ENABLED
00438   #include "stm32l4xx_hal_swpmi.h"
00439 #endif /* HAL_SWPMI_MODULE_ENABLED */
00440
00441 #ifdef HAL_TIM_MODULE_ENABLED
00442   #include "stm32l4xx_hal_tim.h"
00443 #endif /* HAL_TIM_MODULE_ENABLED */
00444
00445 #ifdef HAL_TSC_MODULE_ENABLED
00446   #include "stm32l4xx_hal_tsc.h"
00447 #endif /* HAL_TSC_MODULE_ENABLED */
00448
00449 #ifdef HAL_UART_MODULE_ENABLED
00450   #include "stm32l4xx_hal_uart.h"
00451 #endif /* HAL_UART_MODULE_ENABLED */
00452
00453 #ifdef HAL_USART_MODULE_ENABLED
00454   #include "stm32l4xx_hal_usart.h"
00455 #endif /* HAL_USART_MODULE_ENABLED */
00456
00457 #ifdef HAL_WWDG_MODULE_ENABLED
00458   #include "stm32l4xx_hal_wwdg.h"
00459 #endif /* HAL_WWDG_MODULE_ENABLED */
00460
00461 /* Exported macro ------------------------------------------------------------*/
00462 #ifdef  USE_FULL_ASSERT
00471   #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
00472 /* Exported functions ------------------------------------------------------- */
00473   void assert_failed(uint8_t *file, uint32_t line);
00474 #else
00475   #define assert_param(expr) ((void)0U)
00476 #endif /* USE_FULL_ASSERT */
00477
```

```
00478 #ifdef __cplusplus
00479 }
00480 #endif
00481
00482 #endif /* STM32L4xx_HAL_CONF_H */
```

7.12 Inc/stm32l4xx_it.h File Reference

This file contains the headers of the interrupt handlers.

Functions

- void **NMI_Handler** (void)

  This function handles Non maskable interrupt.
- void **HardFault_Handler** (void)

  This function handles Hard fault interrupt.
- void **MemManage_Handler** (void)

  This function handles Memory management fault.
- void **BusFault_Handler** (void)

  This function handles Prefetch fault, memory access fault.
- void **UsageFault_Handler** (void)

  This function handles Undefined instruction or illegal state.
- void **DebugMon_Handler** (void)

  This function handles Debug monitor.
- void **TIM1_UP_TIM16_IRQHandler** (void)

  This function handles TIM1 update interrupt and TIM16 global interrupt.

7.12.1 Detailed Description

This file contains the headers of the interrupt handlers.

Attention

7.13 stm32l4xx_it.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 /* Define to prevent recursive inclusion -------------------------------------*/
00021 #ifndef __STM32L4xx_IT_H
00022 #define __STM32L4xx_IT_H
00023
00024 #ifdef __cplusplus
00025  extern "C" {
00026 #endif
00027
00028 /* Private includes ----------------------------------------------------------*/
00029 /* USER CODE BEGIN Includes */
00030
00031 /* USER CODE END Includes */
00032
00033 /* Exported types ------------------------------------------------------------*/
00034 /* USER CODE BEGIN ET */
00035
00036 /* USER CODE END ET */
00037
00038 /* Exported constants --------------------------------------------------------*/
00039 /* USER CODE BEGIN EC */
00040
00041 /* USER CODE END EC */
00042
00043 /* Exported macro ------------------------------------------------------------*/
00044 /* USER CODE BEGIN EM */
00045
00046 /* USER CODE END EM */
00047
00048 /* Exported functions prototypes ---------------------------------------------*/
00049 void NMI_Handler(void);
00050 void HardFault_Handler(void);
00051 void MemManage_Handler(void);
00052 void BusFault_Handler(void);
00053 void UsageFault_Handler(void);
00054 void DebugMon_Handler(void);
00055 void TIM1_UP_TIM16_IRQHandler(void);
00056 /* USER CODE BEGIN EFP */
00057
00058 /* USER CODE END EFP */
00059
00060 #ifdef __cplusplus
00061 }
00062 #endif
00063
00064 #endif /* __STM32L4xx_IT_H */
```

7.14 Inc/usart.h File Reference

This file contains all the function prototypes for the usart.c file.

```
#include "main.h"
```

Functions

- void **MX_USART2_UART_Init** (void)

Variables

- UART_HandleTypeDef **huart2**

7.14.1 Detailed Description

This file contains all the function prototypes for the usart.c file.

Attention

7.15 usart.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00019 /* USER CODE END Header */
00020 /* Define to prevent recursive inclusion -------------------------------------*/
00021 #ifndef __USART_H__
00022 #define __USART_H__
00023
00024 #ifdef __cplusplus
00025 extern "C" {
00026 #endif
00027
00028 /* Includes ------------------------------------------------------------------*/
00029 #include "main.h"
00030
00031 /* USER CODE BEGIN Includes */
00032
00033 /* USER CODE END Includes */
00034
00035 extern UART_HandleTypeDef huart2;
00036
00037 /* USER CODE BEGIN Private defines */
00038
00039 /* USER CODE END Private defines */
00040
00041 void MX_USART2_UART_Init(void);
00042
00043 /* USER CODE BEGIN Prototypes */
00044
00045 /* USER CODE END Prototypes */
00046
00047 #ifdef __cplusplus
00048 }
00049 #endif
00050
00051 #endif /* __USART_H__ */
00052
```

7.16 Src/gpio.c File Reference

This file provides code for the configuration of all used GPIO pins.

```
#include "gpio.h"
```

Functions

- void MX_GPIO_Init (void)

7.16.1 Detailed Description

This file provides code for the configuration of all used GPIO pins.

Attention

7.16.2 Function Documentation

7.16.2.1 MX_GPIO_Init()

```
void MX_GPIO_Init (
             void  )
```

Configure pins as Analog Input Output EVENT_OUT EXTI

7.17 Src/lights_functions.c File Reference

Traffic lights, functions for controlling traffic lights.

lights_functions.c.

```
#include "lights_functions.h"
```

Functions

- void feed_shift_reg (uint8_t *lights)

  feed_shift_reg, feed shift register with 24 bit input.
- void format_lights (const lstate_t *curr, uint8_t *lights)

  format_lights, transform state of traffic lights to 24 bit input.
- uint8_t get_buttons ()

  get_buttons, returns an 8-bit integer. Bits 0-5 represent one button each.
- void toggle_blue (light_t *p_light)

  toggle_blue, change logical state of ligh to red if currently blue and blue if it is currently red. Does nothing if state is neither red nor blue.
- **pedestrian_go** (light_t *p_light)

7.17.1 Detailed Description

Traffic lights, functions for controlling traffic lights.

lights_functions.c.

Author

Adam Brolin

Version

1.0

Date

03-December-2024

Functions and structures for controlling and handling flow of traffic lights.

7.17.2 Function Documentation

7.17.2.1 feed_shift_reg()

```
void feed_shift_reg (
            uint8_t * lights )
```

feed_shift_reg, feed shift register with 24 bit input.

Parameters

uint8_t	*lights, pointer to 24 bits divided into 3 bytes. Each bit represent an led on the 74HC595D shield, some bits are unmapped.

Returns

void, no return value

7.17.2.2 format_lights()

```
void format_lights (
            const lstate_t * curr,
            uint8_t * lights )
```

format_lights, transform state of traffic lights to 24 bit input.

Parameters

const	lstate_t ∗curr, the current state of the traffic intersection.
uint8_t	∗lights, location where 24 bit value will return.

Returns

void, no return value.

7.17.2.3 get_buttons()

```
uint8_t get_buttons ( )
```

get_buttons, returns an 8-bit integer. Bits 0-5 represent one button each.

Parameters

void,no	parameter.

Returns

uint8_t, bits 0-5 represent one button each. Use macros BTN_X_XXXX with bit-wise AND to check for specific button. 1 is on, 0 is off.

7.17.2.4 toggle_blue()

```
void toggle_blue (
            light_t * p_light )
```

toggle_blue, change logical state of ligh to red if currently blue and blue if it is currently red. Does nothing if state is neither red nor blue.

Note

Logical blue represents both red and blue led on.

Parameters

light_t	∗p_light, pointer to the logical state of a pedestrian light.

Returns

void, no return value.

7.18 Src/lights_test.c File Reference

Traffic light tests, functions for testing traffic light functions.

lights_test.c.

```
#include "lights_test.h"
#include <stdint.h>
```

Functions

- void **test_program** (lstate_t *state)
- void **test_feed_shift_ref** ()
- **test_format_lights** ()
- void **test_get_buttons** (lstate_t *state)
- void **test_toggle_blue** (lstate_t *state)

7.18.1 Detailed Description

Traffic light tests, functions for testing traffic light functions.

lights_test.c.

Author

Adam Brolin

Version

1.0

Date

08-December-2024

Enable testing program on line 259 in freertos.c

7.19 Src/main.c File Reference

: Main program body

```
#include "main.h"
#include "cmsis_os.h"
#include "spi.h"
#include "usart.h"
#include "gpio.h"
#include "lights_functions.h"
#include "lights_test.h"
```

Functions

- void SystemClock_Config (void)

  System Clock Configuration.
- void MX_FREERTOS_Init (void)

  FreeRTOS initialization.
- int main (void)

  The application entry point.
- void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef *htim)

  Period elapsed callback in non blocking mode.
- void Error_Handler (void)

  This function is executed in case of error occurrence.

7.19.1 Detailed Description

: Main program body

Attention

7.19.2 Function Documentation

7.19.2.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

7.19.2.2 HAL_TIM_PeriodElapsedCallback()

```
void HAL_TIM_PeriodElapsedCallback (
             TIM_HandleTypeDef * htim )
```

Period elapsed callback in non blocking mode.

Note

This function is called when TIM1 interrupt took place, inside HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick() to increment a global variable "uwTick" used as application time base.

Parameters

htim	: TIM handle

Return values

None	

7.19.2.3 main()

```
int main (
             void  )
```

The application entry point.

Return values

int	

7.19.2.4 MX_FREERTOS_Init()

```
void MX_FREERTOS_Init (
             void  )
```

FreeRTOS initialization.

Parameters

None	

Return values

None	

7.19.2.5 SystemClock_Config()

```
void SystemClock_Config (
             void  )
```

System Clock Configuration.

Return values

None	

Configure the main internal regulator output voltage

Initializes the RCC Oscillators according to the specified parameters in the RCC_OscInitTypeDef structure.

Initializes the CPU, AHB and APB buses clocks

7.20 Src/spi.c File Reference

This file provides code for the configuration of the SPI instances.

```
#include "spi.h"
```

Functions

- void **MX_SPI3_Init** (void)
- void HAL_SPI_MspInit (SPI_HandleTypeDef *spiHandle)
- void HAL_SPI_MspDeInit (SPI_HandleTypeDef *spiHandle)

Variables

- SPI_HandleTypeDef **hspi3**

7.20.1 Detailed Description

This file provides code for the configuration of the SPI instances.

Attention

7.20.2 Function Documentation

7.20.2.1 HAL_SPI_MspDeInit()

```
void HAL_SPI_MspDeInit (
            SPI_HandleTypeDef * spiHandle )
```

SPI3 GPIO Configuration PA4 ------> SPI3_NSS PC10 ------> SPI3_SCK PB5 ------> SPI3_MOSI

7.20.2.2 HAL_SPI_MspInit()

```
void HAL_SPI_MspInit (
            SPI_HandleTypeDef * spiHandle )
```

SPI3 GPIO Configuration PA4 ------> SPI3_NSS PC10 ------> SPI3_SCK PB5 ------> SPI3_MOSI

7.21 Src/stm32l4xx_hal_msp.c File Reference

This file provides code for the MSP Initialization and de-Initialization codes.

```
#include "main.h"
```

Functions

- void HAL_MspInit (void)

7.21.1 Detailed Description

This file provides code for the MSP Initialization and de-Initialization codes.

Attention

7.21.2 Function Documentation

7.21.2.1 HAL_MspInit()

```
void HAL_MspInit (
            void  )
```

Initializes the Global MSP.

7.22 Src/stm32l4xx_hal_timebase_tim.c File Reference

HAL time base based on the hardware TIM.

```
#include "stm32l4xx_hal.h"
#include "stm32l4xx_hal_tim.h"
```

Functions

- HAL_StatusTypeDef HAL_InitTick (uint32_t TickPriority)

  This function configures the TIM1 as a time base source. The time source is configured to have 1ms time base with a dedicated Tick interrupt priority.
- void HAL_SuspendTick (void)

  Suspend Tick increment.
- void HAL_ResumeTick (void)

  Resume Tick increment.

Variables

- TIM_HandleTypeDef **htim1**

7.22.1 Detailed Description

HAL time base based on the hardware TIM.

Attention

7.22.2 Function Documentation

7.22.2.1 HAL_InitTick()

```
HAL_StatusTypeDef HAL_InitTick (
            uint32_t TickPriority )
```

This function configures the TIM1 as a time base source. The time source is configured to have 1ms time base with a dedicated Tick interrupt priority.

Note

This function is called automatically at the beginning of program after reset by HAL_Init() or at any time when clock is configured, by HAL_RCC_ClockConfig().

Parameters

TickPriority	Tick interrupt priority.

Return values

HAL	status

7.22.2.2 HAL_ResumeTick()

```
void HAL_ResumeTick (
             void  )
```

Resume Tick increment.

Note

Enable the tick increment by Enabling TIM1 update interrupt.

Parameters

None	

Return values

None	

7.22.2.3 HAL_SuspendTick()

```
void HAL_SuspendTick (
             void  )
```

Suspend Tick increment.

Note

Disable the tick increment by disabling TIM1 update interrupt.

Parameters

None	

Return values

None	

7.23 Src/stm32l4xx_it.c File Reference

Interrupt Service Routines.

```
#include "main.h"
#include "stm32l4xx_it.h"
```

Functions

- void **NMI_Handler** (void)

  This function handles Non maskable interrupt.
- void **HardFault_Handler** (void)

  This function handles Hard fault interrupt.
- void **MemManage_Handler** (void)

  This function handles Memory management fault.
- void **BusFault_Handler** (void)

  This function handles Prefetch fault, memory access fault.
- void **UsageFault_Handler** (void)

  This function handles Undefined instruction or illegal state.
- void **DebugMon_Handler** (void)

  This function handles Debug monitor.
- void **TIM1_UP_TIM16_IRQHandler** (void)

  This function handles TIM1 update interrupt and TIM16 global interrupt.

Variables

- TIM_HandleTypeDef **htim1**

7.23.1 Detailed Description

Interrupt Service Routines.

Attention

7.24 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>
```

Functions

- 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

7.24.1 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

7.25 Src/sysmem.c File Reference

STM32CubeIDE System Memory calls file.

```
#include <errno.h>
#include <stdint.h>
```

Functions

- void * _sbrk (ptrdiff_t incr)

  _sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

7.25.1 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

7.25.2 Function Documentation

7.25.2.1 _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

7.26 Src/system_stm32l4xx.c File Reference

CMSIS Cortex-M4 Device Peripheral Access Layer System Source File.

```
#include "stm32l4xx.h"
```

Macros

- #define HSE_VALUE 8000000U
- #define MSI_VALUE 4000000U
- #define HSI_VALUE 16000000U

Functions

- void SystemInit (void)

  Setup the microcontroller system.
- void SystemCoreClockUpdate (void)

  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Variables

- uint32_t **SystemCoreClock** = 4000000U
- const uint8_t **AHBPrescTable** [16] = {0U, 0U, 0U, 0U, 0U, 0U, 0U, 0U, 1U, 2U, 3U, 4U, 6U, 7U, 8U, 9U}
- const uint8_t **APBPrescTable** [8] = {0U, 0U, 0U, 0U, 1U, 2U, 3U, 4U}
- const uint32_t MSIRangeTable [12]

7.26.1 Detailed Description

CMSIS Cortex-M4 Device Peripheral Access Layer System Source File.

Author

MCD Application Team

This file provides two functions and one global variable to be called from user application:

- SystemInit(): This function is called at startup just after reset and before branch to main program. This call is made inside the "startup_stm32l4xx.s" file.
- SystemCoreClock variable: Contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.
- SystemCoreClockUpdate(): Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.

After each device reset the MSI (4 MHz) is used as system clock source. Then SystemInit() function is called, in "startup_stm32l4xx.s" file, to configure the system clock before to branch to main program.

7.26.2 This file configures the system clock as follows:

7.26.2.1 System Clock source | MSI

7.26.2.2 SYSCLK(Hz) | 4000000

7.26.2.3 HCLK(Hz) | 4000000

7.26.2.4 AHB Prescaler | 1

7.26.2.5 APB1 Prescaler | 1

7.26.2.6 APB2 Prescaler | 1

7.26.2.7 PLL_M | 1

7.26.2.8 PLL_N | 8

7.26.2.9 PLL_P | 7

7.26.2.10 PLL_Q | 2

7.26.2.11 PLL_R | 2

7.26.2.12 PLLSAI1_P | NA

7.26.2.13 PLLSAI1_Q | NA

7.26.2.14 PLLSAI1_R | NA

7.26.2.15 PLLSAI2_P | NA

7.26.2.16 PLLSAI2_Q | NA

7.26.2.17 PLLSAI2_R | NA

Require 48MHz for USB OTG FS, | Disabled

7.26.2.18 SDIO and RNG clock |

==========

Attention

7.27 Src/usart.c File Reference

This file provides code for the configuration of the USART instances.
`#include "usart.h"`

Functions

- void **MX_USART2_UART_Init** (void)
- void HAL_UART_MspInit (UART_HandleTypeDef ∗uartHandle)
- void HAL_UART_MspDeInit (UART_HandleTypeDef ∗uartHandle)

Variables

- UART_HandleTypeDef **huart2**

7.27.1 Detailed Description

This file provides code for the configuration of the USART instances.

Attention

7.27.2 Function Documentation

7.27.2.1 HAL_UART_MspDeInit()

```
void HAL_UART_MspDeInit (
            UART_HandleTypeDef * uartHandle )
```

USART2 GPIO Configuration PA2 ------> USART2_TX PA3 ------> USART2_RX

7.27.2.2 HAL_UART_MspInit()

```
void HAL_UART_MspInit (
            UART_HandleTypeDef * uartHandle )
```

Initializes the peripherals clock
USART2 GPIO Configuration PA2 ------> USART2_TX PA3 ------> USART2_RX