IS1300 Project
HT-2022
KTH

Emil Fredriksson
Thu Dec 15 2022

Table of Contents

Table of contents

Module Index

Modules

Here is a list of all modules:

CMSIS 4
Stm32l4xx_system 5
STM32L4xx_System_Private_Includes 6
STM32L4xx_System_Private_TypesDefinitions 7
STM32L4xx_System_Private_Defines 8
STM32L4xx_System_Private_Macros 9
STM32L4xx_System_Private_Variables 10
STM32L4xx_System_Private_FunctionPrototypes 11
STM32L4xx_System_Private_Functions 12

File Index

File List

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

Inc/FreeRTOSConfig.h 14

Inc/gpio.h (This file contains all the function prototypes for the gpio.c file) 17

Inc/main.h (: Header for main.c file. This file contains the common defines of the application) 19

Inc/spi.h (This file contains all the function prototypes for the spi.c file) 23

Inc/stm32l4xx_hal_conf.h (HAL configuration template file. This file should be copied to the application folder and renamed to stm32l4xx_hal_conf.h) 25

Inc/stm32l4xx_it.h (This file contains the headers of the interrupt handlers) 36

Inc/streetFunc.h (: Header for streetFunc.c file. This file contains the functions used in the project) 38

Inc/Test.h (: Header for Test.c file. This file contains the Test functions used in developing the project) 42

Src/gpio.c (This file provides code for the configuration of all used GPIO pins) 44

Src/main.c (: Main program body) 45

Src/spi.c (This file provides code for the configuration of the SPI instances) 47

Src/stm32l4xx_hal_msp.c (This file provides code for the MSP Initialization and de-Initialization codes) 48

Src/stm32l4xx_hal_timebase_tim.c (HAL time base based on the hardware TIM) 49

Src/stm32l4xx_it.c (Interrupt Service Routines) 51

Src/syscalls.c (STM32CubeIDE Minimal System calls file) 52

Src/sysmem.c (STM32CubeIDE System Memory calls file) 54

Src/system_stm32l4xx.c (CMSIS Cortex-M4 Device Peripheral Access Layer System Source File) 56

Module Documentation

CMSIS

Modules

- **Stm32l4xx_system**

Detailed Description

Stm32l4xx_system

Modules

- **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**

Detailed Description

STM32L4xx_System_Private_Includes

STM32L4xx_System_Private_TypesDefinitions

STM32L4xx_System_Private_Defines

Macros

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

Detailed Description

Macro Definition Documentation

#define HSE_VALUE 8000000U

Value of the External oscillator in Hz

#define HSI_VALUE 16000000U

Value of the Internal oscillator in Hz

#define MSI_VALUE 4000000U

Value of the Internal oscillator in Hz

STM32L4xx_System_Private_Macros

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]

Detailed Description

Variable Documentation

const uint32_t MSIRangeTable[12]

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

STM32L4xx_System_Private_FunctionPrototypes

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.

Detailed Description

Function Documentation

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	

void SystemInit (void)

Setup the microcontroller system.

Return values

None	

File Documentation

FreeRTOSConfig.h

```
1 /* USER CODE BEGIN Header */
2 /*
3  * FreeRTOS Kernel V10.3.1
4  * Portion Copyright (C) 2017 Amazon.com, Inc. or its affiliates.  All Rights Reserved.
5  * Portion Copyright (C) 2019 StMicroelectronics, Inc.  All Rights Reserved.
6  *
7  * Permission is hereby granted, free of charge, to any person obtaining a copy of
8  * this software and associated documentation files (the "Software"), to deal in
9  * the Software without restriction, including without limitation the rights to
10  * use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
11  * the Software, and to permit persons to whom the Software is furnished to do so,
12  * subject to the following conditions:
13  *
14  * The above copyright notice and this permission notice shall be included in all
15  * copies or substantial portions of the Software.
16  *
17  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
18  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
19  * FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
20  * COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
21  * IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
22  * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
23  *
24  * http://www.FreeRTOS.org
25  * http://aws.amazon.com/freertos
26  *
27  * 1 tab == 4 spaces!
28  */
29 /* USER CODE END Header */
30
31 #ifndef FREERTOS CONFIG H
32 #define FREERTOS_CONFIG_H
33
34 /*-----------------------------------------------------------
35  * Application specific definitions.
36  *
37  * These definitions should be adjusted for your particular hardware and
38  * application requirements.
39  *
40  * These parameters and more are described within the 'configuration' section of the
41  * FreeRTOS API documentation available on the FreeRTOS.org web site.
42  *
43  * See http://www.freertos.org/a00110.html
44  *----------------------------------------------------------*/
45
46 /* USER CODE BEGIN Includes */
47 /* Section where include file can be added */
48 /* USER CODE END Includes */
49
50 /* Ensure definitions are only used by the compiler, and not by the assembler. */
51 #if defined(  ICCARM  ) || defined(  CC ARM) || defined(  GNUC  )
52   #include <stdint.h>
53   extern uint32_t SystemCoreClock;
54 #endif
55 #ifndef CMSIS_device_header
56 #define CMSIS_device_header "stm32l4xx.h"
57 #endif /* CMSIS_device_header */
58
59 #define configENABLE_FPU                        0
60 #define configENABLE_MPU                        0
61
62 #define configUSE PREEMPTION                    1
63 #define configSUPPORT_STATIC_ALLOCATION         1
64 #define configSUPPORT_DYNAMIC_ALLOCATION        1
65 #define configUSE_IDLE_HOOK                     0
66 #define configUSE TICK HOOK                     0
67 #define configCPU CLOCK HZ                      ( SystemCoreClock )
68 #define configTICK_RATE_HZ                      ((TickType_t)1000)
69 #define configMAX_PRIORITIES                    ( 56 )
```

```
70 #define configMINIMAL_STACK_SIZE              ((uint16_t)128)
71 #define configTOTAL_HEAP_SIZE                ((size_t)4000)
72 #define configMAX_TASK_NAME_LEN               ( 16 )
73 #define configUSE_TRACE_FACILITY              1
74 #define configUSE_16_BIT_TICKS                0
75 #define configUSE_MUTEXES                     1
76 #define configQUEUE_REGISTRY_SIZE             8
77 #define configUSE_RECURSIVE_MUTEXES           1
78 #define configUSE_COUNTING_SEMAPHORES         1
79 #define configUSE_PORT_OPTIMISED_TASK_SELECTION  0
80 /* USER CODE BEGIN MESSAGE_BUFFER_LENGTH_TYPE */
81 /* Defaults to size_t for backward compatibility, but can be changed
82   if lengths will always be less than the number of bytes in a size_t. */
83 #define configMESSAGE_BUFFER_LENGTH_TYPE       size_t
84 /* USER CODE END MESSAGE_BUFFER_LENGTH_TYPE */
85
86 /* Co-routine definitions. */
87 #define configUSE_CO_ROUTINES                 0
88 #define configMAX_CO_ROUTINE_PRIORITIES         ( 2 )
89
90 /* Software timer definitions. */
91 #define configUSE_TIMERS                      1
92 #define configTIMER_TASK_PRIORITY              ( 2 )
93 #define configTIMER_QUEUE_LENGTH              10
94 #define configTIMER_TASK_STACK_DEPTH           256
95
96 /* The following flag must be enabled only when using newlib */
97 #define configUSE_NEWLIB_REENTRANT        1
98
99 /* CMSIS-RTOS V2 flags */
100 #define configUSE_OS2_THREAD_SUSPEND_RESUME  1
101 #define configUSE_OS2_THREAD_ENUMERATE      1
102 #define configUSE_OS2_EVENTFLAGS_FROM_ISR    1
103 #define configUSE_OS2_THREAD_FLAGS          1
104 #define configUSE_OS2_TIMER                1
105 #define configUSE_OS2_MUTEX                1
106
107 /* Set the following definitions to 1 to include the API function, or zero
108 to exclude the API function. */
109 #define INCLUDE_vTaskPrioritySet           1
110 #define INCLUDE_uxTaskPriorityGet          1
111 #define INCLUDE_vTaskDelete                1
112 #define INCLUDE_vTaskCleanUpResources       0
113 #define INCLUDE_vTaskSuspend               1
114 #define INCLUDE_vTaskDelayUntil            1
115 #define INCLUDE_vTaskDelay                 1
116 #define INCLUDE_xTaskGetSchedulerState      1
117 #define INCLUDE_xTimerPendFunctionCall      1
118 #define INCLUDE_xQueueGetMutexHolder        1
119 #define INCLUDE_uxTaskGetStackHighWaterMark  1
120 #define INCLUDE_xTaskGetCurrentTaskHandle    1
121 #define INCLUDE_eTaskGetState              1
122
123 /*
124  * The CMSIS-RTOS V2 FreeRTOS wrapper is dependent on the heap implementation used
125  * by the application thus the correct define need to be enabled below
126  */
127 #define USE_FreeRTOS_HEAP_4
128
129 /* Cortex-M specific definitions. */
130 #ifdef __NVIC_PRIO_BITS
131  /*   BVIC_PRIO_BITS will be specified when CMSIS is being used. */
132  #define configPRIO_BITS          NVIC_PRIO_BITS
133 #else
134  #define configPRIO_BITS        4
135 #endif
136
137 /* The lowest interrupt priority that can be used in a call to a "set priority"
138 function. */
139 #define configLIBRARY_LOWEST_INTERRUPT_PRIORITY   15
140
141 /* The highest interrupt priority that can be used by any interrupt service
142 routine that makes calls to interrupt safe FreeRTOS API functions.  DO NOT CALL
143 INTERRUPT SAFE FREERTOS API FUNCTIONS FROM ANY INTERRUPT THAT HAS A HIGHER
144 PRIORITY THAN THIS! (higher priorities are lower numeric values. */
145 #define configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY 5
146
```

```
147 /* Interrupt priorities used by the kernel port layer itself.  These are generic
148 to all Cortex-M ports, and do not rely on any particular library functions. */
149 #define configKERNEL INTERRUPT PRIORITY         (
configLIBRARY LOWEST INTERRUPT PRIORITY << (8 - configPRIO BITS) )
150 /* !!!! configMAX_SYSCALL_INTERRUPT_PRIORITY must not be set to zero !!!!
151 See http://www.FreeRTOS.org/RTOS-Cortex-M3-M4.html. */
152 #define configMAX SYSCALL INTERRUPT PRIORITY    (
configLIBRARY MAX SYSCALL INTERRUPT PRIORITY << (8 - configPRIO BITS) )
153
154 /* Normal assert() semantics without relying on the provision of an assert.h
155 header file. */
156 /* USER CODE BEGIN 1 */
157 #define configASSERT( x ) if ((x) == 0) {taskDISABLE_INTERRUPTS(); for( ;; );}
158 /* USER CODE END 1 */
159
160 /* Definitions that map the FreeRTOS port interrupt handlers to their CMSIS
161 standard names. */
162 #define vPortSVCHandler    SVC_Handler
163 #define xPortPendSVHandler PendSV Handler
164
165 /* IMPORTANT: After 10.3.1 update, Systick_Handler comes from NVIC (if SYS timebase
= systick), otherwise from cmsis_os2.c */
166
167 #define USE_CUSTOM_SYSTICK_HANDLER_IMPLEMENTATION 0
168
169 /* USER CODE BEGIN Defines */
170 /* Section where parameter definitions can be added (for instance, to override default
ones in FreeRTOS.h) */
171 /* USER CODE END Defines */
172
173 #endif /* FREERTOS_CONFIG_H */
```

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)

Detailed Description

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

Attention

Function Documentation

void MX_GPIO_Init (void)

Configure pins as Analog Input Output EVENT_OUT EXTI

gpio.h

Go to the documentation of this file.

```
1 /* USER CODE BEGIN Header */
19 /* USER CODE END Header */
20 /* Define to prevent recursive inclusion -------------------------------------*/
21 #ifndef __GPIO_H__
22 #define __GPIO_H__
23
24 #ifdef __cplusplus
25 extern "C" {
26 #endif
27
28 /* Includes ------------------------------------------------------------------*/
29 #include "main.h"
30
31 /* USER CODE BEGIN Includes */
32
33 /* USER CODE END Includes */
34
35 /* USER CODE BEGIN Private defines */
36
37 /* USER CODE END Private defines */
38
39 void MX_GPIO_Init(void);
40
41 /* USER CODE BEGIN Prototypes */
42
43 /* USER CODE END Prototypes */
44
45 #ifdef __cplusplus
46 }
47 #endif
48 #endif /*   GPIO H   */
49
```

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 **TL1_Car_Pin** GPIO_PIN_4
- #define **TL1_Car_GPIO_Port** GPIOC
- #define **STCP_Pin** GPIO_PIN_12
- #define **STCP_GPIO_Port** GPIOB
- #define **TL2_Car_Pin** GPIO_PIN_13
- #define **TL2_Car_GPIO_Port** GPIOB
- #define **TL3_Car_Pin** GPIO_PIN_14
- #define **TL3_Car_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 **TL4_Car_Pin** GPIO_PIN_10
- #define **TL4_Car_GPIO_Port** GPIOA
- #define **PL1_Switch_Pin** GPIO_PIN_15
- #define **PL1_Switch_GPIO_Port** GPIOA
- #define **SCHP_Pin** GPIO_PIN_10
- #define **SCHP_GPIO_Port** GPIOC
- #define **PL2_Switch_Pin** GPIO_PIN_7
- #define **PL2_Switch_GPIO_Port** GPIOB

Functions

- void **Error_Handler** (void)
  This function is executed in case of error occurrence.

Detailed Description

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

Attention

Function Documentation

void Error_Handler (void)

This function is executed in case of error occurrence.

Return values

None	

main.h

Go to the documentation of this file.

```
1 /* USER CODE BEGIN Header */
19 /* USER CODE END Header */
20
21 /* Define to prevent recursive inclusion -------------------------------------*/
22 #ifndef __MAIN_H
23 #define __MAIN_H
24
25 #ifdef __cplusplus
26 extern "C" {
27 #endif
28
29 /* Includes ------------------------------------------------------------------*/
30 #include "stm32l4xx_hal.h"
31
32 /* Private includes ----------------------------------------------------------*/
33 /* USER CODE BEGIN Includes */
34
35 /* USER CODE END Includes */
36
37 /* Exported types ------------------------------------------------------------*/
38 /* USER CODE BEGIN ET */
39
40 /* USER CODE END ET */
41
42 /* Exported constants --------------------------------------------------------*/
43 /* USER CODE BEGIN EC */
44
45 /* USER CODE END EC */
46
47 /* Exported macro ------------------------------------------------------------*/
48 /* USER CODE BEGIN EM */
49
50 /* USER CODE END EM */
51
52 /* Exported functions prototypes ---------------------------------------------*/
53 void Error_Handler(void);
54
55 /* USER CODE BEGIN EFP */
56
57 /* USER CODE END EFP */
58
59 /* Private defines -----------------------------------------------------------*/
60 #define TL1_Car_Pin GPIO_PIN_4
61 #define TL1_Car_GPIO_Port GPIOC
62 #define STCP_Pin GPIO_PIN_12
63 #define STCP_GPIO_Port GPIOB
64 #define TL2_Car_Pin GPIO_PIN_13
65 #define TL2_Car_GPIO_Port GPIOB
66 #define TL3_Car_Pin GPIO_PIN_14
67 #define TL3_Car_GPIO_Port GPIOB
68 #define Enable_Pin GPIO_PIN_7
69 #define Enable_GPIO_Port GPIOC
70 #define Reset_Pin GPIO_PIN_9
71 #define Reset_GPIO_Port GPIOA
72 #define TL4_Car_Pin GPIO_PIN_10
73 #define TL4_Car_GPIO_Port GPIOA
74 #define PL1_Switch_Pin GPIO_PIN_15
75 #define PL1_Switch_GPIO_Port GPIOA
76 #define SCHP_Pin GPIO_PIN_10
77 #define SCHP_GPIO_Port GPIOC
78 #define PL2_Switch_Pin GPIO_PIN_7
79 #define PL2_Switch_GPIO_Port GPIOB
80
81 /* USER CODE BEGIN Private defines */
82
83 /* USER CODE END Private defines */
84
85 #ifdef __cplusplus
86 }
87 #endif
88
89 #endif /* __MAIN_H */
```

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**

Detailed Description

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

Attention

spi.h

Go to the documentation of this file.

```
1 /* USER CODE BEGIN Header */
19 /* USER CODE END Header */
20 /* Define to prevent recursive inclusion -------------------------------------*/
21 #ifndef __SPI_H__
22 #define __SPI_H__
23
24 #ifdef __cplusplus
25 extern "C" {
26 #endif
27
28 /* Includes ------------------------------------------------------------------*/
29 #include "main.h"
30
31 /* USER CODE BEGIN Includes */
32
33 /* USER CODE END Includes */
34
35 extern SPI_HandleTypeDef hspi3;
36
37 /* USER CODE BEGIN Private defines */
38
39 /* USER CODE END Private defines */
40
41 void MX_SPI3_Init(void);
42
43 /* USER CODE BEGIN Prototypes */
44
45 /* USER CODE END Prototypes */
46
47 #ifdef   cplusplus
48 }
49 #endif
50
51 #endif /* __SPI_H__ */
52
```

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_i2c.h"
#include "stm32l4xx_hal_pwr.h"
#include "stm32l4xx_hal_spi.h"
#include "stm32l4xx_hal_tim.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_GPIO_MODULE_ENABLED**
- #define **HAL_EXTI_MODULE_ENABLED**
- #define **HAL_I2C_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.

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

Macro Definition Documentation

#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

#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

#define HSE_STARTUP_TIMEOUT ((uint32_t)100U)

Time out for HSE start up, in ms

#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

#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.

#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

#define LSE_STARTUP_TIMEOUT 5000U

Time out for LSE start up, in ms

#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

#define LSI_VALUE 32000U

Internal Low Speed oscillator (LSI) value.

LSI Typical Value in Hz

#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

#define TICK_INT_PRIORITY 15U

tick interrupt priority

#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).

#define VDD_VALUE 3300U

This is the HAL system configuration section.

Value of VDD in mv

stm32l4xx_hal_conf.h

Go to the documentation of this file.

```
1 /* USER CODE BEGIN Header */
21 /* USER CODE END Header */
22
23 /* Define to prevent recursive inclusion -------------------------------------*/
24 #ifndef STM32L4xx_HAL_CONF_H
25 #define STM32L4xx_HAL_CONF_H
26
27 #ifdef __cplusplus
28  extern "C" {
29 #endif
30
31 /* Exported types ------------------------------------------------------------*/
32 /* Exported constants --------------------------------------------------------*/
33
34 /* ########################## Module Selection ############################## */
38 #define HAL_MODULE_ENABLED
39 /*#define HAL_ADC_MODULE_ENABLED   */
40 /*#define HAL_CRYP_MODULE_ENABLED   */
41 /*#define HAL_CAN_MODULE_ENABLED   */
42 /*#define HAL_COMP_MODULE_ENABLED   */
43 /*#define HAL_CRC_MODULE_ENABLED   */
44 /*#define HAL_CRYP_MODULE_ENABLED   */
45 /*#define HAL_DAC_MODULE_ENABLED   */
46 /*#define HAL_DCMI_MODULE_ENABLED   */
47 /*#define HAL_DMA2D_MODULE_ENABLED   */
48 /*#define HAL_DFSDM_MODULE_ENABLED   */
49 /*#define HAL_DSI_MODULE_ENABLED   */
50 /*#define HAL_FIREWALL_MODULE_ENABLED   */
51 /*#define HAL_GFXMMU_MODULE_ENABLED   */
52 /*#define HAL_HCD_MODULE_ENABLED   */
53 /*#define HAL_HASH_MODULE_ENABLED   */
54 /*#define HAL_I2S_MODULE_ENABLED   */
55 /*#define HAL_IRDA_MODULE_ENABLED   */
56 /*#define HAL_IWDG_MODULE_ENABLED   */
57 /*#define HAL_LTDC_MODULE_ENABLED   */
58 /*#define HAL_LCD_MODULE_ENABLED   */
59 /*#define HAL_LPTIM_MODULE_ENABLED   */
60 /*#define HAL_MMC_MODULE_ENABLED   */
61 /*#define HAL_NAND_MODULE_ENABLED   */
62 /*#define HAL_NOR_MODULE_ENABLED   */
63 /*#define HAL_OPAMP_MODULE_ENABLED   */
64 /*#define HAL_OSPI_MODULE_ENABLED   */
65 /*#define HAL_OSPI_MODULE_ENABLED   */
66 /*#define HAL_PCD_MODULE_ENABLED   */
67 /*#define HAL_PKA_MODULE_ENABLED   */
68 /*#define HAL_QSPI_MODULE_ENABLED   */
69 /*#define HAL_QSPI_MODULE_ENABLED   */
70 /*#define HAL_RNG_MODULE_ENABLED   */
71 /*#define HAL_RTC_MODULE_ENABLED   */
72 /*#define HAL_SAI_MODULE_ENABLED   */
73 /*#define HAL_SD_MODULE_ENABLED   */
74 /*#define HAL_SMBUS_MODULE_ENABLED   */
75 /*#define HAL_SMARTCARD_MODULE_ENABLED   */
76 #define HAL_SPI_MODULE_ENABLED
77 /*#define HAL_SRAM_MODULE_ENABLED   */
78 /*#define HAL_SWPMI_MODULE_ENABLED   */
79 #define HAL_TIM_MODULE_ENABLED
80 /*#define HAL_TSC_MODULE_ENABLED   */
81 /*#define HAL_UART_MODULE_ENABLED   */
82 /*#define HAL_USART_MODULE_ENABLED   */
83 /*#define HAL_WWDG_MODULE_ENABLED   */
84 /*#define HAL_EXTI_MODULE_ENABLED   */
85 /*#define HAL_PSSI_MODULE_ENABLED   */
86 #define HAL_GPIO_MODULE_ENABLED
87 #define HAL_EXTI_MODULE_ENABLED
88 #define HAL_I2C_MODULE_ENABLED
89 #define HAL_DMA_MODULE_ENABLED
90 #define HAL_RCC_MODULE_ENABLED
91 #define HAL_FLASH_MODULE_ENABLED
92 #define HAL_PWR_MODULE_ENABLED
93 #define HAL_CORTEX_MODULE_ENABLED
94
95 /* ########################## Oscillator Values adaptation ####################*/
```

```
101 #if !defined  (HSE_VALUE)
102   #define HSE_VALUE    ((uint32_t)8000000U)
103 #endif /* HSE VALUE */
104
105 #if !defined  (HSE_STARTUP_TIMEOUT)
106   #define HSE_STARTUP_TIMEOUT    ((uint32_t)100U)
107 #endif /* HSE STARTUP TIMEOUT */
108
113 #if !defined  (MSI VALUE)
114   #define MSI_VALUE    ((uint32_t)4000000U)
115 #endif /* MSI_VALUE */
121 #if !defined  (HSI_VALUE)
122   #define HSI_VALUE    ((uint32_t)16000000U)
123 #endif /* HSI_VALUE */
124
132 #if !defined  (HSI48_VALUE)
133  #define HSI48_VALUE   ((uint32_t)48000000U)
135 #endif /* HSI48_VALUE */
136
140 #if !defined  (LSI VALUE)
141  #define LSI_VALUE  32000U
142 #endif /* LSI_VALUE */
150 #if !defined  (LSE_VALUE)
151   #define LSE_VALUE    32768U
152 #endif /* LSE_VALUE */
153
154 #if !defined  (LSE_STARTUP_TIMEOUT)
155   #define LSE_STARTUP_TIMEOUT    5000U
156 #endif /* HSE_STARTUP_TIMEOUT */
157
163 #if !defined  (EXTERNAL SAI1 CLOCK VALUE)
164   #define EXTERNAL SAI1 CLOCK VALUE    2097000U
165 #endif /* EXTERNAL_SAI1_CLOCK_VALUE */
166
172 #if !defined  (EXTERNAL SAI2 CLOCK VALUE)
173   #define EXTERNAL SAI2 CLOCK VALUE    2097000U
174 #endif /* EXTERNAL SAI2 CLOCK VALUE */
175
176 /* Tip: To avoid modifying this file each time you need to use different HSE,
177    ===  you can define the HSE value in your toolchain compiler preprocessor. */
178
179 /* ########################### System Configuration ######################### */
184 #define  VDD_VALUE                  3300U
185 #define  TICK_INT_PRIORITY           15U
186 #define  USE_RTOS                   0U
187 #define  PREFETCH ENABLE             1U
188 #define  INSTRUCTION CACHE ENABLE     1U
189 #define  DATA_CACHE_ENABLE           1U
190
191 /* ########################## Assert Selection ############################## */
196 /* #define USE_FULL_ASSERT    1U */
197
198 /* ################### Register callback feature configuration ############### */
208 #define USE_HAL_ADC_REGISTER_CALLBACKS       0U
209 #define USE_HAL_CAN_REGISTER_CALLBACKS       0U
210 #define USE_HAL_COMP_REGISTER_CALLBACKS      0U
211 #define USE_HAL_CRYP_REGISTER_CALLBACKS      0U
212 #define USE HAL DAC REGISTER CALLBACKS       0U
213 #define USE HAL DCMI REGISTER CALLBACKS      0U
214 #define USE_HAL_DFSDM_REGISTER_CALLBACKS     0U
215 #define USE_HAL_DMA2D_REGISTER_CALLBACKS     0U
216 #define USE HAL DSI REGISTER CALLBACKS       0U
217 #define USE HAL GFXMMU REGISTER CALLBACKS    0U
218 #define USE_HAL_HASH_REGISTER_CALLBACKS      0U
219 #define USE_HAL_HCD_REGISTER_CALLBACKS       0U
220 #define USE_HAL_I2C_REGISTER_CALLBACKS       0U
221 #define USE_HAL_IRDA_REGISTER_CALLBACKS      0U
222 #define USE_HAL_LPTIM_REGISTER_CALLBACKS     0U
223 #define USE_HAL_LTDC_REGISTER_CALLBACKS      0U
224 #define USE_HAL_MMC_REGISTER_CALLBACKS       0U
225 #define USE_HAL_OPAMP_REGISTER_CALLBACKS     0U
226 #define USE_HAL_OSPI_REGISTER_CALLBACKS      0U
227 #define USE_HAL_PCD_REGISTER_CALLBACKS       0U
228 #define USE HAL QSPI REGISTER CALLBACKS      0U
229 #define USE HAL RNG REGISTER CALLBACKS       0U
230 #define USE_HAL_RTC_REGISTER_CALLBACKS       0U
231 #define USE_HAL_SAI_REGISTER_CALLBACKS       0U
```

```
232 #define USE_HAL_SD_REGISTER_CALLBACKS        0U
233 #define USE_HAL_SMARTCARD_REGISTER_CALLBACKS  0U
234 #define USE_HAL_SMBUS_REGISTER_CALLBACKS      0U
235 #define USE_HAL_SPI_REGISTER_CALLBACKS        0U
236 #define USE_HAL_SWPMI_REGISTER_CALLBACKS      0U
237 #define USE_HAL_TIM_REGISTER_CALLBACKS        0U
238 #define USE_HAL_TSC_REGISTER_CALLBACKS        0U
239 #define USE_HAL_UART_REGISTER_CALLBACKS       0U
240 #define USE_HAL_USART_REGISTER_CALLBACKS      0U
241 #define USE_HAL_WWDG_REGISTER_CALLBACKS       0U
242
243 /* ################## SPI peripheral configuration ########################## */
244
245 /* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver
246  * Activated: CRC code is present inside driver
247  * Deactivated: CRC code cleaned from driver
248  */
249
250 #define USE_SPI_CRC                  0U
251
252 /* Includes ------------------------------------------------------------------*/
257 #ifdef HAL_RCC_MODULE_ENABLED
258   #include "stm32l4xx_hal_rcc.h"
259 #endif /* HAL_RCC_MODULE_ENABLED */
260
261 #ifdef HAL_GPIO_MODULE_ENABLED
262   #include "stm32l4xx_hal_gpio.h"
263 #endif /* HAL_GPIO_MODULE_ENABLED */
264
265 #ifdef HAL_DMA_MODULE_ENABLED
266   #include "stm32l4xx_hal_dma.h"
267 #endif /* HAL_DMA_MODULE_ENABLED */
268
269 #ifdef HAL_DFSDM_MODULE_ENABLED
270   #include "stm32l4xx_hal_dfsdm.h"
271 #endif /* HAL_DFSDM_MODULE_ENABLED */
272
273 #ifdef HAL_CORTEX_MODULE_ENABLED
274   #include "stm32l4xx_hal_cortex.h"
275 #endif /* HAL_CORTEX_MODULE_ENABLED */
276
277 #ifdef HAL_ADC_MODULE_ENABLED
278   #include "stm32l4xx_hal_adc.h"
279 #endif /* HAL_ADC_MODULE_ENABLED */
280
281 #ifdef HAL_CAN_MODULE_ENABLED
282   #include "stm32l4xx_hal_can.h"
283 #endif /* HAL_CAN_MODULE_ENABLED */
284
285 #ifdef HAL_CAN_LEGACY_MODULE_ENABLED
286   #include "Legacy/stm32l4xx_hal_can_legacy.h"
287 #endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
288
289 #ifdef HAL_COMP_MODULE_ENABLED
290   #include "stm32l4xx_hal_comp.h"
291 #endif /* HAL_COMP_MODULE_ENABLED */
292
293 #ifdef HAL_CRC_MODULE_ENABLED
294   #include "stm32l4xx_hal_crc.h"
295 #endif /* HAL_CRC_MODULE_ENABLED */
296
297 #ifdef HAL_CRYP_MODULE_ENABLED
298   #include "stm32l4xx_hal_cryp.h"
299 #endif /* HAL_CRYP_MODULE_ENABLED */
300
301 #ifdef HAL_DAC_MODULE_ENABLED
302   #include "stm32l4xx_hal_dac.h"
303 #endif /* HAL_DAC_MODULE_ENABLED */
304
305 #ifdef HAL_DCMI_MODULE_ENABLED
306   #include "stm32l4xx_hal_dcmi.h"
307 #endif /* HAL_DCMI_MODULE_ENABLED */
308
309 #ifdef HAL_DMA2D_MODULE_ENABLED
310   #include "stm32l4xx_hal_dma2d.h"
311 #endif /* HAL_DMA2D_MODULE_ENABLED */
312
```

```
313 #ifdef HAL_DSI_MODULE_ENABLED
314   #include "stm32l4xx_hal_dsi.h"
315 #endif /* HAL DSI MODULE ENABLED */
316
317 #ifdef HAL_EXTI_MODULE_ENABLED
318   #include "stm32l4xx_hal_exti.h"
319 #endif /* HAL EXTI MODULE ENABLED */
320
321 #ifdef HAL GFXMMU MODULE ENABLED
322   #include "stm32l4xx_hal_gfxmmu.h"
323 #endif /* HAL_GFXMMU_MODULE_ENABLED */
324
325 #ifdef HAL_FIREWALL_MODULE_ENABLED
326   #include "stm32l4xx_hal_firewall.h"
327 #endif /* HAL FIREWALL MODULE ENABLED */
328
329 #ifdef HAL_FLASH_MODULE_ENABLED
330   #include "stm32l4xx_hal_flash.h"
331 #endif /* HAL FLASH MODULE ENABLED */
332
333 #ifdef HAL_HASH_MODULE_ENABLED
334   #include "stm32l4xx_hal_hash.h"
335 #endif /* HAL_HASH_MODULE_ENABLED */
336
337 #ifdef HAL_HCD_MODULE_ENABLED
338   #include "stm32l4xx_hal_hcd.h"
339 #endif /* HAL_HCD_MODULE_ENABLED */
340
341 #ifdef HAL_I2C_MODULE_ENABLED
342   #include "stm32l4xx_hal_i2c.h"
343 #endif /* HAL I2C MODULE ENABLED */
344
345 #ifdef HAL_IRDA_MODULE_ENABLED
346   #include "stm32l4xx_hal_irda.h"
347 #endif /* HAL IRDA MODULE ENABLED */
348
349 #ifdef HAL IWDG MODULE ENABLED
350   #include "stm32l4xx_hal_iwdg.h"
351 #endif /* HAL_IWDG_MODULE_ENABLED */
352
353 #ifdef HAL_LCD_MODULE_ENABLED
354   #include "stm32l4xx_hal_lcd.h"
355 #endif /* HAL_LCD_MODULE_ENABLED */
356
357 #ifdef HAL_LPTIM_MODULE_ENABLED
358   #include "stm32l4xx hal lptim.h"
359 #endif /* HAL LPTIM MODULE ENABLED */
360
361 #ifdef HAL_LTDC_MODULE_ENABLED
362   #include "stm32l4xx_hal_ltdc.h"
363 #endif /* HAL_LTDC_MODULE_ENABLED */
364
365 #ifdef HAL_MMC_MODULE_ENABLED
366   #include "stm32l4xx_hal_mmc.h"
367 #endif /* HAL_MMC_MODULE_ENABLED */
368
369 #ifdef HAL_NAND_MODULE_ENABLED
370   #include "stm32l4xx hal nand.h"
371 #endif /* HAL NAND MODULE ENABLED */
372
373 #ifdef HAL_NOR_MODULE_ENABLED
374   #include "stm32l4xx hal nor.h"
375 #endif /* HAL NOR MODULE ENABLED */
376
377 #ifdef HAL_OPAMP_MODULE_ENABLED
378   #include "stm32l4xx_hal_opamp.h"
379 #endif /* HAL_OPAMP_MODULE_ENABLED */
380
381 #ifdef HAL_OSPI_MODULE_ENABLED
382   #include "stm32l4xx_hal_ospi.h"
383 #endif /* HAL_OSPI_MODULE_ENABLED */
384
385 #ifdef HAL_PCD_MODULE_ENABLED
386   #include "stm32l4xx hal pcd.h"
387 #endif /* HAL PCD MODULE ENABLED */
388
389 #ifdef HAL_PKA_MODULE_ENABLED
```

```
390   #include "stm32l4xx_hal_pka.h"
391 #endif /* HAL_PKA_MODULE_ENABLED */
392
393 #ifdef HAL_PSSI_MODULE_ENABLED
394   #include "stm32l4xx_hal_pssi.h"
395 #endif /* HAL_PSSI_MODULE_ENABLED */
396
397 #ifdef HAL_PWR_MODULE_ENABLED
398   #include "stm32l4xx_hal_pwr.h"
399 #endif /* HAL_PWR_MODULE_ENABLED */
400
401 #ifdef HAL_QSPI_MODULE_ENABLED
402   #include "stm32l4xx_hal_qspi.h"
403 #endif /* HAL_QSPI_MODULE_ENABLED */
404
405 #ifdef HAL_RNG_MODULE_ENABLED
406   #include "stm32l4xx_hal_rng.h"
407 #endif /* HAL_RNG_MODULE_ENABLED */
408
409 #ifdef HAL_RTC_MODULE_ENABLED
410   #include "stm32l4xx_hal_rtc.h"
411 #endif /* HAL_RTC_MODULE_ENABLED */
412
413 #ifdef HAL_SAI_MODULE_ENABLED
414   #include "stm32l4xx_hal_sai.h"
415 #endif /* HAL_SAI_MODULE_ENABLED */
416
417 #ifdef HAL_SD_MODULE_ENABLED
418   #include "stm32l4xx_hal_sd.h"
419 #endif /* HAL_SD_MODULE_ENABLED */
420
421 #ifdef HAL_SMARTCARD_MODULE_ENABLED
422   #include "stm32l4xx_hal_smartcard.h"
423 #endif /* HAL_SMARTCARD_MODULE_ENABLED */
424
425 #ifdef HAL_SMBUS_MODULE_ENABLED
426   #include "stm32l4xx_hal_smbus.h"
427 #endif /* HAL_SMBUS_MODULE_ENABLED */
428
429 #ifdef HAL_SPI_MODULE_ENABLED
430   #include "stm32l4xx_hal_spi.h"
431 #endif /* HAL_SPI_MODULE_ENABLED */
432
433 #ifdef HAL_SRAM_MODULE_ENABLED
434   #include "stm32l4xx_hal_sram.h"
435 #endif /* HAL_SRAM_MODULE_ENABLED */
436
437 #ifdef HAL_SWPMI_MODULE_ENABLED
438   #include "stm32l4xx_hal_swpmi.h"
439 #endif /* HAL_SWPMI_MODULE_ENABLED */
440
441 #ifdef HAL_TIM_MODULE_ENABLED
442   #include "stm32l4xx_hal_tim.h"
443 #endif /* HAL_TIM_MODULE_ENABLED */
444
445 #ifdef HAL_TSC_MODULE_ENABLED
446   #include "stm32l4xx_hal_tsc.h"
447 #endif /* HAL_TSC_MODULE_ENABLED */
448
449 #ifdef HAL_UART_MODULE_ENABLED
450   #include "stm32l4xx_hal_uart.h"
451 #endif /* HAL_UART_MODULE_ENABLED */
452
453 #ifdef HAL_USART_MODULE_ENABLED
454   #include "stm32l4xx_hal_usart.h"
455 #endif /* HAL_USART_MODULE_ENABLED */
456
457 #ifdef HAL_WWDG_MODULE_ENABLED
458   #include "stm32l4xx_hal_wwdg.h"
459 #endif /* HAL_WWDG_MODULE_ENABLED */
460
461 /* Exported macro ------------------------------------------------------------*/
462 #ifdef  USE_FULL_ASSERT
471   #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)  __FILE__ ,
  __LINE__ ))
472 /* Exported functions ------------------------------------------------------- */
473   void assert_failed(uint8_t *file, uint32_t line);
```

```
474 #else
475   #define assert_param(expr) ((void)0U)
476 #endif /* USE_FULL_ASSERT */
477
478 #ifdef __cplusplus
479 }
480 #endif
481
482 #endif /* STM32L4xx_HAL_CONF_H */
```

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.

Detailed Description

This file contains the headers of the interrupt handlers.

Attention

stm32l4xx_it.h

Go to the documentation of this file.

```
1 /* USER CODE BEGIN Header */
18 /* USER CODE END Header */
19
20 /* Define to prevent recursive inclusion -------------------------------------*/
21 #ifndef __STM32L4xx_IT_H
22 #define __STM32L4xx_IT_H
23
24 #ifdef __cplusplus
25  extern "C" {
26 #endif
27
28 /* Private includes ----------------------------------------------------------*/
29 /* USER CODE BEGIN Includes */
30
31 /* USER CODE END Includes */
32
33 /* Exported types ------------------------------------------------------------*/
34 /* USER CODE BEGIN ET */
35
36 /* USER CODE END ET */
37
38 /* Exported constants --------------------------------------------------------*/
39 /* USER CODE BEGIN EC */
40
41 /* USER CODE END EC */
42
43 /* Exported macro ------------------------------------------------------------*/
44 /* USER CODE BEGIN EM */
45
46 /* USER CODE END EM */
47
48 /* Exported functions prototypes ---------------------------------------------*/
49 void NMI_Handler(void);
50 void HardFault_Handler(void);
51 void MemManage_Handler(void);
52 void BusFault_Handler(void);
53 void UsageFault_Handler(void);
54 void DebugMon_Handler(void);
55 void TIM1_UP_TIM16_IRQHandler(void);
56 /* USER CODE BEGIN EFP */
57
58 /* USER CODE END EFP */
59
60 #ifdef __cplusplus
61 }
62 #endif
63
64 #endif /* __STM32L4xx_IT_H */
```

Inc/streetFunc.h File Reference

: Header for streetFunc.c file. This file contains the functions used in the project

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

Functions

- void **Led_Clear** (void)
- void **Blue_Clear1** (void)
  Function to clear all LEDS Led_Clear();.

- void **Blue_Clear2** (void)
  Function to clear the blue light on Crossing 1 Blue_Clear();.

- void **Led_Toggle** (uint8_t n, char state)
  Function to clear the blue light on Crossing 2 Blue_Clear();.

- void **Flash_Led** (uint8_t n, char state, uint8_t delay, uint8_t times)
  Function light up a specific colour of a specific traffic/crossing light Led_Toggle(n, 'state');.

- void **States** (uint8_t state)
  Function to toogle on/off LED in a loop Flash_Led(n, 'state', delay, times);.

- bool **Crossing_Button1** (void)
  Function containing the different states of the crossing and the functions to show each of them.

- bool **Crossing_Button2** (void)
  Function for checking if button 1 has been pressed.

- bool **Car_Horizontal** (void)
  Function for checking if button 2 has been pressed.

- bool **Car_Vertical** (void)
  Function for checking if any cars are present on the horizontal road.

- void **sendSPIdata** (uint8_t data[], uint8_t byteSize)
  Function for checking if any cars are present on the vertical road.

Detailed Description

: Header for streetFunc.c file. This file contains the functions used in the project

Attention

Function Documentation

void Blue_Clear1 (void)

Function to clear all LEDS Led_Clear();.

Parameters

void	

void Blue_Clear2 (void)

Function to clear the blue light on Crossing 1 Blue_Clear();.

Parameters

void	

bool Car_Horizontal (void)

Function for checking if button 2 has been pressed.

void **Crossing_Button2()**;

Parameters

void	

Returns

bool, true if Button 2 has been pressed, otherwise false

bool Car_Vertical (void)

Function for checking if any cars are present on the horizontal road.

Car_Horizontal(void);

Parameters

void	

Returns

bool, true if Car is present, otherwise false

bool Crossing_Button1 (void)

Function containing the different states of the crossing and the functions to show each of them.

void

state = 1: Vertical cars have green light

state = 2: Horizontal cars have green light

state = 3: Yellow lights are displayed States(state);

Template Parameters

uint8_t	state: Choosing which state is active

bool Crossing_Button2 (void)

Function for checking if button 1 has been pressed.

void **Crossing_Button2()**;

Parameters

void	

Returns

bool, true if Button 1 has been pressed, otherwise false

void Flash_Led (uint8_t *n*, char *state*, uint8_t *delay*, uint8_t *times*)

Function light up a specific colour of a specific traffic/crossing light Led_Toggle(n, 'state');.

Parameters

1	uint8_t n : Which traffic light is to be toggled
2	char state : Which colour is to be toggled

void Led_Toggle (uint8_t *n*, char *state*)

Function to clear the blue light on Crossing 2 Blue_Clear();.

Parameters

void	

void sendSPIdata (uint8_t *data*[], uint8_t *byteSize*)

Function for checking if any cars are present on the vertical road.

Car_Vertical();

Parameters

void	

void States (uint8_t *state*)

Function to toogle on/off LED in a loop Flash_Led(n, 'state', delay, times);.

Parameters

1	uint8_t delay: Ms delay between blink
2	uint8_t times: How many times LED should blink

streetFunc.h

Go to the documentation of this file.

```
1 /* USER CODE BEGIN Header */
19 /* USER CODE END Header */
20
21 #ifndef INC_STREETFUNC_C_
22 #define INC_STREETFUNC_C_
23 #include <stdbool.h>
24 #include <stdint.h>
25
26
27
28 #endif /* INC_STREETFUNC_C_ */
29
30
31
32 void Led_Clear(void);
37 void Blue_Clear1(void);
42 void Blue_Clear2(void);
47
48 void Led_Toggle(uint8_t n, char state);
53
54 void Flash_Led(uint8_t n, char state, uint8_t delay, uint8_t times);
59
60 void States(uint8_t state);
70
71 bool Crossing Button1(void);
78
79 bool Crossing_Button2(void);
86
87 bool Car Horizontal(void);
93
94 bool Car Vertical(void);
99 // /@return bool, true if Car is present, otherwise false
100
101 void sendSPIdata(uint8_t data[], uint8_t byteSize);
```

Inc/Test.h File Reference

: Header for Test.c file. This file contains the Test functions used in developing the project.

Functions

- void **Test_Led** (void)
- void **Test_program** (void)

Detailed Description

: Header for Test.c file. This file contains the Test functions used in developing the project.

Attention

Test.h

Go to the documentation of this file.

```
1 /* USER CODE BEGIN Header */
19 /* USER CODE END Header */
20
21 #ifndef SRC_TEST_H_
22 #define SRC_TEST_H_
23
24
25
26 #endif /* SRC TEST H  */
27 void Test Led(void);
28 void Test_program(void);
```

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)

Detailed Description

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

Attention

Function Documentation

void MX_GPIO_Init (void)

Configure pins as Analog Input Output EVENT_OUT EXTI

Src/main.c File Reference

: Main program body

```
#include "main.h"
#include "cmsis_os.h"
#include "spi.h"
#include "gpio.h"
#include "FreeRTOS.h"
#include "streetFunc.H"
#include "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.

Detailed Description

: Main program body

Attention

Function Documentation

void Error_Handler (void)

This function is executed in case of error occurrence.

Return values

None	

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	

int main (void)

The application entry point.

Return values

int	

void MX_FREERTOS_Init (void)

FreeRTOS initialization.

Parameters

None	

Return values

None	

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

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**

Detailed Description

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

Attention

Function Documentation

void HAL_SPI_MspDeInit (SPI_HandleTypeDef * *spiHandle*)

SPI3 GPIO Configuration PC10 ---—> SPI3_SCK PB5 ---—> SPI3_MOSI

void HAL_SPI_MspInit (SPI_HandleTypeDef * *spiHandle*)

SPI3 GPIO Configuration PC10 ---—> SPI3_SCK PB5 ---—> SPI3_MOSI

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)

Detailed Description

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

Attention

Function Documentation

void HAL_MspInit (void)

Initializes the Global MSP.

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**

Detailed Description

HAL time base based on the hardware TIM.

Attention

Function Documentation

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

void HAL_ResumeTick (void)

Resume Tick increment.

Note

Enable the tick increment by Enabling TIM1 update interrupt.

Parameters

None	

Return values

None	

void HAL_SuspendTick (void)

Suspend Tick increment.

Note

Disable the tick increment by disabling TIM1 update interrupt.

Parameters

None	

Return values

None	

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**

Detailed Description

Interrupt Service Routines.

Attention

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

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

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

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*

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

Function Documentation

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

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]

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.

This file configures the system clock as follows:

System Clock source	**MSI**
SYSCLK(Hz)	**4000000**
HCLK(Hz)	**4000000**
AHB Prescaler	**1**
APB1 Prescaler	**1**
APB2 Prescaler	**1**
PLL_M	**1**
PLL_N	**8**
PLL_P	**7**
PLL_Q	**2**
PLL_R	**2**
PLLSAI1_P	**NA**
PLLSAI1_Q	**NA**
PLLSAI1_R	**NA**
PLLSAI2_P	**NA**

PLLSAI2_Q **| NA**

PLLSAI2_R **| NA**

Require 48MHz for USB OTG FS, | Disabled

SDIO and RNG clock **|**

==========

Attention

Index

INDEX