PMIK

Generated by Doxygen 1.8.17

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Module Documentation

3.1 CMSIS

Modules

- Stm32l1xx_system

3.1.1 Detailed Description

3.2 Stm32l1xx_system

Modules

- STM32L1xx_System_Private_Includes
- STM32L1xx_System_Private_TypesDefinitions
- STM32L1xx_System_Private_Defines
- STM32L1xx_System_Private_Macros
- STM32L1xx_System_Private_Variables
- STM32L1xx_System_Private_FunctionPrototypes
- STM32L1xx_System_Private_Functions

3.2.1 Detailed Description

3.3 STM32L1xx_System_Private_Includes

3.4 STM32L1xx_System_Private_TypesDefinitions

3.5 STM32L1xx_System_Private_Defines

Macros

- #define HSE_VALUE ((uint32_t)8000000U)
- #define HSI_VALUE ((uint32_t)8000000U)
- #define VECT_TAB_OFFSET 0x00U

3.5.1 Detailed Description

3.5.2 Macro Definition Documentation

3.5.2.1 HSE_VALUE

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

Default value of the External oscillator in Hz. This value can be provided and adapted by the user application.

3.5.2.2 HSI_VALUE

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

Default value of the Internal oscillator in Hz. This value can be provided and adapted by the user application.

3.5.2.3 VECT_TAB_OFFSET

```
#define VECT_TAB_OFFSET 0x00U
```

< Uncomment the following line if you need to use external SRAM mounted on STM32L152D_EVAL board as data memory

< Uncomment the following line if you need to relocate your vector Table in Internal SRAM. Vector Table base offset field. This value must be a multiple of 0x200.

3.6 STM32L1xx_System_Private_Macros

3.7 STM32L1xx_System_Private_Variables

Variables

- uint32_t SystemCoreClock = 2097000U
- const uint8_t PLLMulTable [9] = {3U, 4U, 6U, 8U, 12U, 16U, 24U, 32U, 48U}
- 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}

3.7.1 Detailed Description

3.7.2 Variable Documentation

3.7.2.1 AHBPrescTable

```
const uint8_t AHBPrescTable[16] = {0U, 0U, 0U, 0U, 0U, 0U, 0U, 0U, 1U, 2U, 3U, 4U, 6U, 7U, 8U,
9U}
```

3.7.2.2 APBPrescTable

```
const uint8_t APBPrescTable[8] = {0U, 0U, 0U, 0U, 1U, 2U, 3U, 4U}
```

3.7.2.3 PLLMulTable

```
const uint8_t PLLMulTable[9] = {3U, 4U, 6U, 8U, 12U, 16U, 24U, 32U, 48U}
```

3.7.2.4 SystemCoreClock

```
uint32_t SystemCoreClock = 2097000U
```

3.8 STM32L1xx_System_Private_FunctionPrototypes

3.9 STM32L1xx_System_Private_Functions

Functions

- void SystemInit (void)

  Setup the microcontroller system. Initialize the Embedded Flash Interface, the PLL and update the SystemCoreClock variable.
- void SystemCoreClockUpdate (void)

  Update SystemCoreClock 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.

3.9.1 Detailed Description

3.9.2 Function Documentation

3.9.2.1 SystemCoreClockUpdate()

```
void SystemCoreClockUpdate (
            void  )
```

Update SystemCoreClock 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 as defined by the MSI range.
- 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(∗) multiplied/divided by the PLL factors.

(∗) HSI_VALUE is a constant defined in stm32l1xx.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 stm32l1xx.h file (default value 8 MHz), user has to ensure that HSE_V↩ ALUE 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.

Parameters

None	

Return values

None	

3.9.2.2 SystemInit()

```
void SystemInit (
             void  )
```

Setup the microcontroller system. Initialize the Embedded Flash Interface, the PLL and update the SystemCore↩ Clock variable.

Parameters

None	

Return values

None	

< Set MSION bit

< Reset SW[1:0], HPRE[3:0], PPRE1[2:0], PPRE2[2:0], MCOSEL[2:0] and MCOPRE[2:0] bits

< Reset HSION, HSEON, CSSON and PLLON bits

< Reset HSEBYP bit

< Reset PLLSRC, PLLMUL[3:0] and PLLDIV[1:0] bits

< Disable all interrupts

Chapter 4

File Documentation

4.1 adc.c File Reference

```
#include "adc.h"
```

Functions

- void MX_ADC_Init (void)
- void HAL_ADC_MspInit (ADC_HandleTypeDef *adcHandle)
- void HAL_ADC_MspDeInit (ADC_HandleTypeDef *adcHandle)

Variables

- ADC_HandleTypeDef hadc

4.1.1 Function Documentation

4.1.1.1 HAL_ADC_MspDeInit()

```
void HAL_ADC_MspDeInit (
            ADC_HandleTypeDef * adcHandle )
```

ADC GPIO Configuration
PA0-WKUP1 ------> ADC_IN0

4.1.1.2 HAL_ADC_MspInit()

```
void HAL_ADC_MspInit (
            ADC_HandleTypeDef * adcHandle )
```

ADC GPIO Configuration
PA0-WKUP1 ------> ADC_IN0

4.1.1.3 MX_ADC_Init()

```
void MX_ADC_Init (
             void  )
```

Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of conversion)

Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.

4.1.2 Variable Documentation

4.1.2.1 hadc

```
ADC_HandleTypeDef hadc
```

File Name : ADC.c Description : This file provides code for the configuration of the ADC instances.

Attention

4.2 adc.h File Reference

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

Functions

- void MX_ADC_Init (void)

Variables

- ADC_HandleTypeDef hadc

4.2.1 Function Documentation

4.2.1.1 MX_ADC_Init()

```
void MX_ADC_Init (
             void  )
```

Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of conversion)

Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.

4.2.2 Variable Documentation

4.2.2.1 hadc

```
ADC_HandleTypeDef hadc
```

File Name : ADC.h Description : This file provides code for the configuration of the ADC instances.

Attention

File Name : ADC.c Description : This file provides code for the configuration of the ADC instances.

Attention

4.3 fotodetector.c File Reference

```
#include "fotodetector.h"
```

Functions

- uint8_t get_value_fotodetector ()

4.3.1 Function Documentation

4.3.1.1 get_value_fotodetector()

```
uint8_t get_value_fotodetector ( )
```

File Name : fotodetector.c Description : This file provides code for the configuration of the fotodetector.

4.4 fotodetector.h File Reference

```
#include "main.h"
#include "adc.h"
#include "usart.h"
```

Functions

- uint8_t get_value_fotodetector ()

Variables

- uint8_t fotodetector_value

4.4.1 Function Documentation

4.4.1.1 get_value_fotodetector()

```
uint8_t get_value_fotodetector ( )
```

function for get valude from fotodetector

File Name : fotodetector.c Description : This file provides code for the configuration of the fotodetector.

4.4.2 Variable Documentation

4.4.2.1 fotodetector_value

```
uint8_t fotodetector_value
```

File Name : fotodetector.h Description : This file provides code for the configuration of the fotodetector value taken from fotodetector

4.5 gpio.c File Reference

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

Functions

- void MX_GPIO_Init (void)

4.5.1 Function Documentation

4.5.1.1 MX_GPIO_Init()

```
void MX_GPIO_Init (
             void  )
```

File Name : gpio.c Description : This file provides code for the configuration of all used GPIO pins.

Attention

4.6 gpio.h File Reference

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

Functions

- void MX_GPIO_Init (void)

4.6.1 Function Documentation

4.6.1.1 MX_GPIO_Init()

```
void MX_GPIO_Init (
            void  )
```

File Name : gpio.h Description : This file contains all the functions prototypes for the gpio

Attention

File Name : gpio.c Description : This file provides code for the configuration of all used GPIO pins.

Attention

4.7 main.c File Reference

: Main program body

```
#include "main.h"
#include "adc.h"
#include "usart.h"
#include "gpio.h"
```

Functions

- void SystemClock_Config (void)

  System Clock Configuration.
- int main (void)

  The application entry point.
- void HAL_UART_RxCpltCallback (UART_HandleTypeDef ∗huart)
- void Error_Handler (void)

  This function is executed in case of error occurrence.

4.7.1 Detailed Description

: Main program body

Attention

4.7.2 Function Documentation

4.7.2.1 Error_Handler()

```
void Error_Handler (
            void  )
```

This function is executed in case of error occurrence.

Return values

None	

4.7.2.2 HAL_UART_RxCpltCallback()

```
void HAL_UART_RxCpltCallback (
            UART_HandleTypeDef * huart )
```

uart take command raspberry uart

admin uart

4.7.2.3 main()

```
int main (
            void  )
```

The application entry point.

Return values

int	

wait for uart command from raspberry huart1 is in exit mode

start adc read

wait for uart command from admin huart2 is in exit mode

4.7.2.4 SystemClock_Config()

```
void SystemClock_Config (
            void  )
```

System Clock Configuration.

Return values

None	

Configure the main internal regulator output voltage

Initializes the CPU, AHB and APB busses clocks

Initializes the CPU, AHB and APB busses clocks

4.8 main.h File Reference

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

```
#include "stm32l1xx_hal.h"
```

Macros

- #define B1_Pin GPIO_PIN_13
- #define B1_GPIO_Port GPIOC
- #define B1_EXTI_IRQn EXTI15_10_IRQn
- #define FOTODETECTOR_Pin GPIO_PIN_0
- #define FOTODETECTOR_GPIO_Port GPIOA
- #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 ENABLE_Pin GPIO_PIN_15
- #define ENABLE_GPIO_Port GPIOB
- #define END_DOWN_Pin GPIO_PIN_6
- #define END_DOWN_GPIO_Port GPIOC
- #define END_DOWN_EXTI_IRQn EXTI9_5_IRQn
- #define STEP_Pin GPIO_PIN_7
- #define STEP_GPIO_Port GPIOC
- #define DIR_Pin GPIO_PIN_8
- #define DIR_GPIO_Port GPIOC
- #define END_HIGH_Pin GPIO_PIN_9
- #define END_HIGH_GPIO_Port GPIOC
- #define END_HIGH_EXTI_IRQn EXTI9_5_IRQn
- #define TMS_Pin GPIO_PIN_13
- #define TMS_GPIO_Port GPIOA
- #define TCK_Pin GPIO_PIN_14
- #define TCK_GPIO_Port GPIOA
- #define SWO_Pin GPIO_PIN_3
- #define SWO_GPIO_Port GPIOB

Functions

- void Error_Handler (void)

  This function is executed in case of error occurrence.

Variables

- uint8_t data
- uint8_t able

4.8.1 Detailed Description

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

Attention

4.8.2 Macro Definition Documentation

4.8.2.1 B1_EXTI_IRQn

```
#define B1_EXTI_IRQn EXTI15_10_IRQn
```

4.8.2.2 B1_GPIO_Port

```
#define B1_GPIO_Port GPIOC
```

4.8.2.3 B1_Pin

```
#define B1_Pin GPIO_PIN_13
```

4.8.2.4 DIR_GPIO_Port

```
#define DIR_GPIO_Port GPIOC
```

4.8.2.5 DIR_Pin

```
#define DIR_Pin GPIO_PIN_8
```

4.8.2.6 ENABLE_GPIO_Port

```
#define ENABLE_GPIO_Port GPIOB
```

4.8.2.7 ENABLE_Pin

```
#define ENABLE_Pin GPIO_PIN_15
```

4.8.2.8 END_DOWN_EXTI_IRQn

```
#define END_DOWN_EXTI_IRQn EXTI9_5_IRQn
```

4.8.2.9 END_DOWN_GPIO_Port

```
#define END_DOWN_GPIO_Port GPIOC
```

4.8.2.10 END_DOWN_Pin

```
#define END_DOWN_Pin GPIO_PIN_6
```

4.8.2.11 END_HIGH_EXTI_IRQn

```
#define END_HIGH_EXTI_IRQn EXTI9_5_IRQn
```

4.8.2.12 END_HIGH_GPIO_Port

```
#define END_HIGH_GPIO_Port GPIOC
```

4.8.2.13 END_HIGH_Pin

```
#define END_HIGH_Pin GPIO_PIN_9
```

4.8.2.14 FOTODETECTOR_GPIO_Port

```
#define FOTODETECTOR_GPIO_Port GPIOA
```

4.8.2.15 FOTODETECTOR_Pin

```
#define FOTODETECTOR_Pin GPIO_PIN_0
```

4.8.2.16 LD2_GPIO_Port

```
#define LD2_GPIO_Port GPIOA
```

4.8.2.17 LD2_Pin

```
#define LD2_Pin GPIO_PIN_5
```

4.8.2.18 STEP_GPIO_Port

```
#define STEP_GPIO_Port GPIOC
```

4.8.2.19 STEP_Pin

```
#define STEP_Pin GPIO_PIN_7
```

4.8.2.20 SWO_GPIO_Port

```
#define SWO_GPIO_Port GPIOB
```

4.8.2.21 SWO_Pin

```
#define SWO_Pin GPIO_PIN_3
```

4.8.2.22 TCK_GPIO_Port

```
#define TCK_GPIO_Port GPIOA
```

4.8.2.23 TCK_Pin

```
#define TCK_Pin GPIO_PIN_14
```

4.8.2.24 TMS_GPIO_Port

```
#define TMS_GPIO_Port GPIOA
```

4.8.2.25 TMS_Pin

```
#define TMS_Pin GPIO_PIN_13
```

4.8.2.26 USART_RX_GPIO_Port

```
#define USART_RX_GPIO_Port GPIOA
```

4.8.2.27 USART_RX_Pin

```
#define USART_RX_Pin GPIO_PIN_3
```

4.8.2.28 USART_TX_GPIO_Port

```
#define USART_TX_GPIO_Port GPIOA
```

4.8.2.29 USART_TX_Pin

```
#define USART_TX_Pin GPIO_PIN_2
```

4.8.3 Function Documentation

4.8.3.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

4.8.4 Variable Documentation

4.8.4.1 able

```
uint8_t able
```

data from admin

4.8.4.2 data

```
uint8_t data
```

data from raspberry

4.9 motor.c File Reference

```
#include "motor.h"
```

Functions

- void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)
- void new_data (uint8_t new)
- void admin_pc (uint8_t able)
- void move_motor ()
- void home_up ()
- void home_down ()
- void auto_fotodetector (uint8_t auto_foto)

4.9.1 Function Documentation

4.9.1.1 admin_pc()

```
void admin_pc (
            uint8_t able )
```

work permit

4.9.1.2 auto_fotodetector()

```
void auto_fotodetector (
            uint8_t auto_foto )
```

function to execution comand from fotodetector

4.9.1.3 HAL_GPIO_EXTI_Callback()

```
void HAL_GPIO_EXTI_Callback (
            uint16_t GPIO_Pin )
```

File Name : motor.c Description : This file provides code for the configuration of the motor. function for execute end command

4.9.1.4 home_down()

```
void home_down ( )
```

homeing down blind function

4.9.1.5 home_up()

```
void home_up ( )
```

homeing up blind function

4.9.1.6 move_motor()

```
void move_motor ( )
```

function that sends commands to the engine

4.9.1.7 new_data()

```
void new_data (
            uint8_t new )
```

function to execution comand from uart

4.10 motor.h File Reference

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

Functions

- void move_motor ()
- void new_data (uint8_t new)
- void home_up ()
- void home_down ()
- void auto_fotodetector (uint8_t auto_foto)
- void admin_pc (uint8_t able)

Variables

- uint8_t set_position
- uint8_t position
- uint8_t control
- int dir
- int count_position = 0
- int auto_down
- int auto_up
- uint8_t admin = 1

4.10.1 Function Documentation

4.10.1.1 admin_pc()

```
void admin_pc (
            uint8_t able )
```

work permit

4.10.1.2 auto_fotodetector()

```
void auto_fotodetector (
            uint8_t auto_foto )
```

function to execution comand from fotodetector

4.10.1.3 home_down()

```
void home_down ( )
```

homeing down blind function

4.10.1.4 home_up()

```
void home_up ( )
```

homeing up blind function

4.10.1.5 move_motor()

```
void move_motor ( )
```

function that sends commands to the engine

4.10.1.6 new_data()

```
void new_data (
            uint8_t new )
```

function to execution comand from uart

4.10.2 Variable Documentation

4.10.2.1 admin

```
uint8_t admin = 1
```

work permit

4.10.2.2 auto_down

```
int auto_down
```

motor can go down in auto mode

4.10.2.3 auto_up

```
int auto_up
```

motor can go up in auto mode

4.10.2.4 control

```
uint8_t control
```

value to chosse auto mode

4.10.2.5 count_position

```
int count_position = 0
```

helper variable to include the number of steps in one iteration

4.10.2.6 dir

```
int dir
```

4.10.2.7 position

```
uint8_t position
```

actual position

4.10.2.8 set_position

```
uint8_t set_position
```

File Name : motor.h Description : This file provides code for the configuration of the motor value for setting position

4.11 stm32l1xx_hal_conf.h File Reference

HAL configuration file.

```
#include "stm32l1xx_hal_rcc.h"
#include "stm32l1xx_hal_gpio.h"
#include "stm32l1xx_hal_dma.h"
#include "stm32l1xx_hal_cortex.h"
#include "stm32l1xx_hal_adc.h"
#include "stm32l1xx_hal_flash.h"
#include "stm32l1xx_hal_pwr.h"
#include "stm32l1xx_hal_uart.h"
```

Macros

- #define HAL_MODULE_ENABLED

  This is the list of modules to be used in the HAL driver.
- #define HAL_ADC_MODULE_ENABLED
- #define HAL_UART_MODULE_ENABLED
- #define HAL_GPIO_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)8000000)

  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)100)
- #define MSI_VALUE ((uint32_t)16000000)

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

  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 LSI_VALUE (37000U)

  Internal Low Speed oscillator (LSI) value.
- #define LSE_VALUE ((uint32_t)32768)

  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 ((uint32_t)5000)
- #define VDD_VALUE ((uint32_t)3300)

  This is the HAL system configuration section.
- #define TICK_INT_PRIORITY ((uint32_t)0)
- #define USE_RTOS 0
- #define PREFETCH_ENABLE 0
- #define INSTRUCTION_CACHE_ENABLE 1
- #define DATA_CACHE_ENABLE 1
- #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_COMP_REGISTER_CALLBACKS 0U
- #define USE_HAL_DAC_REGISTER_CALLBACKS 0U
- #define USE_HAL_I2C_REGISTER_CALLBACKS 0U
- #define USE_HAL_I2S_REGISTER_CALLBACKS 0U
- #define USE_HAL_IRDA_REGISTER_CALLBACKS 0U
- #define USE_HAL_OPAMP_REGISTER_CALLBACKS 0U
- #define USE_HAL_PCD_REGISTER_CALLBACKS 0U
- #define USE_HAL_RTC_REGISTER_CALLBACKS 0U
- #define USE_HAL_SDMMC_REGISTER_CALLBACKS 0U
- #define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U
- #define USE_HAL_SPI_REGISTER_CALLBACKS 0U
- #define USE_HAL_TIM_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.

4.11.1 Detailed Description

HAL configuration file.

Attention

4.11.2 Macro Definition Documentation

4.11.2.1 assert_param

```
#define assert_param(
             expr ) ((void)0U)
```

Include module's header file.

4.11.2.2 DATA_CACHE_ENABLE

```
#define DATA_CACHE_ENABLE 1
```

4.11.2.3 HAL_ADC_MODULE_ENABLED

```
#define HAL_ADC_MODULE_ENABLED
```

4.11.2.4 HAL_CORTEX_MODULE_ENABLED

```
#define HAL_CORTEX_MODULE_ENABLED
```

4.11.2.5 HAL_DMA_MODULE_ENABLED

```
#define HAL_DMA_MODULE_ENABLED
```

4.11.2.6 HAL_FLASH_MODULE_ENABLED

```
#define HAL_FLASH_MODULE_ENABLED
```

4.11.2.7 HAL_GPIO_MODULE_ENABLED

```
#define HAL_GPIO_MODULE_ENABLED
```

4.11.2.8 HAL_MODULE_ENABLED

```
#define HAL_MODULE_ENABLED
```

This is the list of modules to be used in the HAL driver.

4.11.2.9 HAL_PWR_MODULE_ENABLED

```
#define HAL_PWR_MODULE_ENABLED
```

4.11.2.10 HAL_RCC_MODULE_ENABLED

```
#define HAL_RCC_MODULE_ENABLED
```

4.11.2.11 HAL_UART_MODULE_ENABLED

```
#define HAL_UART_MODULE_ENABLED
```

4.11.2.12 HSE_STARTUP_TIMEOUT

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

Time out for HSE start up, in ms

4.11.2.13 HSE_VALUE

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

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

4.11.2.14 HSI_VALUE

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

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

4.11.2.15 INSTRUCTION_CACHE_ENABLE

```
#define INSTRUCTION_CACHE_ENABLE 1
```

4.11.2.16 LSE_STARTUP_TIMEOUT

```
#define LSE_STARTUP_TIMEOUT ((uint32_t)5000)
```

Time out for LSE start up, in ms

4.11.2.17 LSE_VALUE

```
#define LSE_VALUE ((uint32_t)32768)
```

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

4.11.2.18 LSI_VALUE

```
#define LSI_VALUE (37000U)
```

Internal Low Speed oscillator (LSI) value.

LSI Typical Value in Hz

4.11.2.19 MSI_VALUE

```
#define MSI_VALUE ((uint32_t)16000000)
```

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

Value of the Internal oscillator in Hz

4.11.2.20 PREFETCH_ENABLE

```
#define PREFETCH_ENABLE 0
```

4.11.2.21 TICK_INT_PRIORITY

```
#define TICK_INT_PRIORITY ((uint32_t)0)
```

tick interrupt priority

4.11.2.22 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 stm32l0xx_hal_ppp.h file for possible callback identifiers defined in HAL_PPP_CallbackIDTypeDef for each PPP peripheral).

4.11.2.23 USE_HAL_COMP_REGISTER_CALLBACKS

```
#define USE_HAL_COMP_REGISTER_CALLBACKS 0U
```

4.11.2.24 USE_HAL_DAC_REGISTER_CALLBACKS

```
#define USE_HAL_DAC_REGISTER_CALLBACKS 0U
```

4.11.2.25 USE_HAL_I2C_REGISTER_CALLBACKS

```
#define USE_HAL_I2C_REGISTER_CALLBACKS 0U
```

4.11.2.26 USE_HAL_I2S_REGISTER_CALLBACKS

```
#define USE_HAL_I2S_REGISTER_CALLBACKS 0U
```

4.11.2.27 USE_HAL_IRDA_REGISTER_CALLBACKS

```
#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U
```

4.11.2.28 USE_HAL_OPAMP_REGISTER_CALLBACKS

```
#define USE_HAL_OPAMP_REGISTER_CALLBACKS 0U
```

4.11.2.29 USE_HAL_PCD_REGISTER_CALLBACKS

```
#define USE_HAL_PCD_REGISTER_CALLBACKS 0U
```

4.11.2.30 USE_HAL_RTC_REGISTER_CALLBACKS

```
#define USE_HAL_RTC_REGISTER_CALLBACKS 0U
```

4.11.2.31 USE_HAL_SDMMC_REGISTER_CALLBACKS

```
#define USE_HAL_SDMMC_REGISTER_CALLBACKS 0U
```

4.11.2.32 USE_HAL_SMARTCARD_REGISTER_CALLBACKS

```
#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U
```

4.11.2.33 USE_HAL_SPI_REGISTER_CALLBACKS

```
#define USE_HAL_SPI_REGISTER_CALLBACKS 0U
```

4.11.2.34 USE_HAL_TIM_REGISTER_CALLBACKS

```
#define USE_HAL_TIM_REGISTER_CALLBACKS 0U
```

4.11.2.35 USE_HAL_UART_REGISTER_CALLBACKS

```
#define USE_HAL_UART_REGISTER_CALLBACKS 0U
```

4.11.2.36 USE_HAL_USART_REGISTER_CALLBACKS

```
#define USE_HAL_USART_REGISTER_CALLBACKS 0U
```

4.11.2.37 USE_HAL_WWDG_REGISTER_CALLBACKS

```
#define USE_HAL_WWDG_REGISTER_CALLBACKS 0U
```

4.11.2.38 USE_RTOS

```
#define USE_RTOS 0
```

4.11.2.39 USE_SPI_CRC

```
#define USE_SPI_CRC 0U
```

4.11.2.40 VDD_VALUE

```
#define VDD_VALUE ((uint32_t)3300)
```

This is the HAL system configuration section.

Value of VDD in mv

4.12 stm32l1xx_hal_msp.c File Reference

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

Functions

- void HAL_MspInit (void)

4.12.1 Function Documentation

4.12.1.1 HAL_MspInit()

```
void HAL_MspInit (
             void  )
```

File Name : stm32l1xx_hal_msp.c Description : This file provides code for the MSP Initialization and de-Initialization codes.

Attention

4.13 stm32l1xx_it.c File Reference

Interrupt Service Routines.

```
#include "main.h"
#include "stm32l1xx_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 Pre-fetch fault, memory access fault.
- void UsageFault_Handler (void)

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

  This function handles System service call via SWI instruction.
- void DebugMon_Handler (void)

  This function handles Debug monitor.
- void PendSV_Handler (void)

  This function handles Pendable request for system service.
- void SysTick_Handler (void)

  This function handles System tick timer.
- void ADC1_IRQHandler (void)

  This function handles ADC global interrupt.
- void EXTI9_5_IRQHandler (void)

  This function handles EXTI line[9:5] interrupts.
- void USART1_IRQHandler (void)

  This function handles USART1 global interrupt.
- void USART2_IRQHandler (void)

  This function handles USART2 global interrupt.
- void USART3_IRQHandler (void)

  This function handles USART3 global interrupt.
- void EXTI15_10_IRQHandler (void)

  This function handles EXTI line[15:10] interrupts.

Variables

- ADC_HandleTypeDef hadc
- UART_HandleTypeDef huart1
- UART_HandleTypeDef huart2
- UART_HandleTypeDef huart3

4.13.1 Detailed Description

Interrupt Service Routines.

Attention

4.13.2 Function Documentation

4.13.2.1 ADC1_IRQHandler()

```
void ADC1_IRQHandler (
             void  )
```

This function handles ADC global interrupt.

4.13.2.2 BusFault_Handler()

```
void BusFault_Handler (
             void  )
```

This function handles Pre-fetch fault, memory access fault.

4.13.2.3 DebugMon_Handler()

```
void DebugMon_Handler (
             void  )
```

This function handles Debug monitor.

4.13.2.4 EXTI15_10_IRQHandler()

```
void EXTI15_10_IRQHandler (
             void  )
```

This function handles EXTI line[15:10] interrupts.

4.13.2.5 EXTI9_5_IRQHandler()

```
void EXTI9_5_IRQHandler (
             void  )
```

This function handles EXTI line[9:5] interrupts.

4.13.2.6 HardFault_Handler()

```
void HardFault_Handler (
             void  )
```

This function handles Hard fault interrupt.

4.13.2.7 MemManage_Handler()

```
void MemManage_Handler (
             void  )
```

This function handles Memory management fault.

4.13.2.8 NMI_Handler()

```
void NMI_Handler (
             void  )
```

This function handles Non maskable interrupt.

4.13.2.9 PendSV_Handler()

```
void PendSV_Handler (
            void  )
```

This function handles Pendable request for system service.

4.13.2.10 SVC_Handler()

```
void SVC_Handler (
            void  )
```

This function handles System service call via SWI instruction.

4.13.2.11 SysTick_Handler()

```
void SysTick_Handler (
            void  )
```

This function handles System tick timer.

4.13.2.12 UsageFault_Handler()

```
void UsageFault_Handler (
            void  )
```

This function handles Undefined instruction or illegal state.

4.13.2.13 USART1_IRQHandler()

```
void USART1_IRQHandler (
            void  )
```

This function handles USART1 global interrupt.

4.13.2.14 USART2_IRQHandler()

```
void USART2_IRQHandler (
            void  )
```

This function handles USART2 global interrupt.

4.13.2.15 USART3_IRQHandler()

```
void USART3_IRQHandler (
            void  )
```

This function handles USART3 global interrupt.

4.13.3 Variable Documentation

4.13.3.1 hadc

```
ADC_HandleTypeDef hadc
```

File Name : ADC.c Description : This file provides code for the configuration of the ADC instances.

Attention

4.13.3.2 huart1

```
UART_HandleTypeDef huart1
```

File Name : USART.c Description : This file provides code for the configuration of the USART instances.

Attention

4.13.3.3 huart2

```
UART_HandleTypeDef huart2
```

4.13.3.4 huart3

```
UART_HandleTypeDef huart3
```

4.14 stm32l1xx_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 Pre-fetch fault, memory access fault.
- void UsageFault_Handler (void)

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

  This function handles System service call via SWI instruction.
- void DebugMon_Handler (void)

  This function handles Debug monitor.
- void PendSV_Handler (void)

  This function handles Pendable request for system service.
- void SysTick_Handler (void)

  This function handles System tick timer.
- void ADC1_IRQHandler (void)

  This function handles ADC global interrupt.
- void EXTI9_5_IRQHandler (void)

  This function handles EXTI line[9:5] interrupts.
- void USART1_IRQHandler (void)

  This function handles USART1 global interrupt.
- void USART2_IRQHandler (void)

  This function handles USART2 global interrupt.
- void USART3_IRQHandler (void)

  This function handles USART3 global interrupt.
- void EXTI15_10_IRQHandler (void)

  This function handles EXTI line[15:10] interrupts.

4.14.1 Detailed Description

This file contains the headers of the interrupt handlers.

Attention

4.14.2 Function Documentation

4.14.2.1 ADC1_IRQHandler()

```
void ADC1_IRQHandler (
             void  )
```

This function handles ADC global interrupt.

4.14.2.2 BusFault_Handler()

```
void BusFault_Handler (
             void  )
```

This function handles Pre-fetch fault, memory access fault.

4.14.2.3 DebugMon_Handler()

```
void DebugMon_Handler (
             void  )
```

This function handles Debug monitor.

4.14.2.4 EXTI15_10_IRQHandler()

```
void EXTI15_10_IRQHandler (
            void  )
```

This function handles EXTI line[15:10] interrupts.

4.14.2.5 EXTI9_5_IRQHandler()

```
void EXTI9_5_IRQHandler (
            void  )
```

This function handles EXTI line[9:5] interrupts.

4.14.2.6 HardFault_Handler()

```
void HardFault_Handler (
            void  )
```

This function handles Hard fault interrupt.

4.14.2.7 MemManage_Handler()

```
void MemManage_Handler (
            void  )
```

This function handles Memory management fault.

4.14.2.8 NMI_Handler()

```
void NMI_Handler (
            void  )
```

This function handles Non maskable interrupt.

4.14.2.9 PendSV_Handler()

```
void PendSV_Handler (
            void  )
```

This function handles Pendable request for system service.

4.14.2.10 SVC_Handler()

```
void SVC_Handler (
             void  )
```

This function handles System service call via SWI instruction.

4.14.2.11 SysTick_Handler()

```
void SysTick_Handler (
             void  )
```

This function handles System tick timer.

4.14.2.12 UsageFault_Handler()

```
void UsageFault_Handler (
             void  )
```

This function handles Undefined instruction or illegal state.

4.14.2.13 USART1_IRQHandler()

```
void USART1_IRQHandler (
             void  )
```

This function handles USART1 global interrupt.

4.14.2.14 USART2_IRQHandler()

```
void USART2_IRQHandler (
             void  )
```

This function handles USART2 global interrupt.

4.14.2.15 USART3_IRQHandler()

```
void USART3_IRQHandler (
             void  )
```

This function handles USART3 global interrupt.

4.15 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

- int errno
- char ** environ = __env

4.15.1 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

4.15.2 Function Documentation

4.15.2.1 __attribute__()

```
__attribute__ (
             (weak)  )
```

4.15.2.2 __io_getchar()

```
int __io_getchar (
             void  )
```

4.15.2.3 __io_putchar()

```
int __io_putchar (
             int ch )
```

4.15.2.4 _close()

```
int _close (
             int file )
```

4.15.2.5 _execve()

```
int _execve (
             char * name,
             char ** argv,
             char ** env )
```

4.15.2.6 _exit()

```
void _exit (
            int status )
```

4.15.2.7 _fork()

```
int _fork (
            void  )
```

4.15.2.8 _fstat()

```
int _fstat (
            int file,
            struct stat * st )
```

4.15.2.9 _getpid()

```
int _getpid (
            void  )
```

4.15.2.10 _isatty()

```
int _isatty (
            int file )
```

4.15.2.11 _kill()

```
int _kill (
            int pid,
            int sig )
```

4.15.2.12 _link()

```
int _link (
              char * old,
              char * new )
```

4.15.2.13 _lseek()

```
int _lseek (
              int file,
              int ptr,
              int dir )
```

4.15.2.14 _open()

```
int _open (
              char * path,
              int flags,
               ...  )
```

4.15.2.15 _stat()

```
int _stat (
              char * file,
              struct stat * st )
```

4.15.2.16 _times()

```
int _times (
              struct tms * buf )
```

4.15.2.17 _unlink()

```
int _unlink (
              char * name )
```

4.15.2.18 _wait()

```
int _wait (
              int * status )
```

4.15.2.19 initialise_monitor_handles()

```
void initialise_monitor_handles ( )
```

4.15.3 Variable Documentation

4.15.3.1 environ

```
char** environ = __env
```

4.15.3.2 errno

```
int errno
```

4.16 sysmem.c File Reference

STM32CubeIDE Minimal System Memory calls file.

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

Functions

- register char *stack_ptr asm ("sp")
- caddr_t _sbrk (int incr)

Variables

- int errno

4.16.1 Detailed Description

STM32CubeIDE Minimal System Memory 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

4.16.2 Function Documentation

4.16.2.1 _sbrk()

```
caddr_t _sbrk (
            int incr )
```

_sbrk Increase program data space. Malloc and related functions depend on this

4.16.2.2 asm()

```
register char* stack_ptr asm (
            "sp"  )
```

4.16.3 Variable Documentation

4.16.3.1 errno

```
int errno
```

4.17 system_stm32l1xx.c File Reference

CMSIS Cortex-M3 Device Peripheral Access Layer System Source File.

```
#include "stm32l1xx.h"
```

Macros

- #define HSE_VALUE ((uint32_t)8000000U)
- #define HSI_VALUE ((uint32_t)8000000U)
- #define VECT_TAB_OFFSET 0x00U

Functions

- void SystemInit (void)

  Setup the microcontroller system. Initialize the Embedded Flash Interface, the PLL and update the SystemCoreClock variable.
- void SystemCoreClockUpdate (void)

  Update SystemCoreClock 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 = 2097000U
- const uint8_t PLLMulTable [9] = {3U, 4U, 6U, 8U, 12U, 16U, 24U, 32U, 48U}
- 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}

4.17.1 Detailed Description

CMSIS Cortex-M3 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_stm32l1xx.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.

Attention

4.18 usart.c File Reference

```
#include "usart.h"
```

Functions

- void MX_USART1_UART_Init (void)
- void MX_USART2_UART_Init (void)
- void MX_USART3_UART_Init (void)
- void HAL_UART_MspInit (UART_HandleTypeDef ∗uartHandle)
- void HAL_UART_MspDeInit (UART_HandleTypeDef ∗uartHandle)

Variables

- UART_HandleTypeDef huart1
- UART_HandleTypeDef huart2
- UART_HandleTypeDef huart3

4.18.1 Function Documentation

4.18.1.1 HAL_UART_MspDeInit()

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

USART1 GPIO Configuration
PA9 ------> USART1_TX PA10 ------> USART1_RX

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

USART3 GPIO Configuration
PB10 ------> USART3_TX PB11 ------> USART3_RX

4.18.1.2 HAL_UART_MspInit()

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

USART1 GPIO Configuration
PA9 ------> USART1_TX PA10 ------> USART1_RX

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

USART3 GPIO Configuration
PB10 ------> USART3_TX PB11 ------> USART3_RX

4.18.1.3 MX_USART1_UART_Init()

```
void MX_USART1_UART_Init (
            void  )
```

4.18.1.4 MX_USART2_UART_Init()

```
void MX_USART2_UART_Init (
            void  )
```

4.18.1.5 MX_USART3_UART_Init()

```
void MX_USART3_UART_Init (
            void  )
```

4.18.2 Variable Documentation

4.18.2.1 huart1

```
UART_HandleTypeDef huart1
```

File Name : USART.c Description : This file provides code for the configuration of the USART instances.

Attention

4.18.2.2 huart2

```
UART_HandleTypeDef huart2
```

4.18.2.3 huart3

```
UART_HandleTypeDef huart3
```

4.19 usart.h File Reference

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

Functions

- void MX_USART1_UART_Init (void)
- void MX_USART2_UART_Init (void)
- void MX_USART3_UART_Init (void)

Variables

- UART_HandleTypeDef huart1
- UART_HandleTypeDef huart2
- UART_HandleTypeDef huart3

4.19.1 Function Documentation

4.19.1.1 MX_USART1_UART_Init()

```
void MX_USART1_UART_Init (
            void  )
```

4.19.1.2 MX_USART2_UART_Init()

```
void MX_USART2_UART_Init (
             void  )
```

4.19.1.3 MX_USART3_UART_Init()

```
void MX_USART3_UART_Init (
             void  )
```

4.19.2 Variable Documentation

4.19.2.1 huart1

```
UART_HandleTypeDef huart1
```

File Name : USART.h Description : This file provides code for the configuration of the USART instances.

Attention

File Name : USART.c Description : This file provides code for the configuration of the USART instances.

Attention

4.19.2.2 huart2

```
UART_HandleTypeDef huart2
```

4.19.2.3 huart3

```
UART_HandleTypeDef huart3
```

Index