

IS1300 Project

Generated by Doxygen 1.9.1

1 Module Index	1
1.1 Modules	1
2 File Index	3
2.1 File List	3
3 Module Documentation	5
3.1 CMSIS	5
3.1.1 Detailed Description	5
3.2 Stm32l4xx_system	5
3.2.1 Detailed Description	5
3.3 STM32L4xx_System_Private_Includes	5
3.4 STM32L4xx_System_Private_TypesDefinitions	5
3.5 STM32L4xx_System_Private_Defines	5
3.5.1 Detailed Description	6
3.5.2 Macro Definition Documentation	6
3.5.2.1 HSE_VALUE	6
3.5.2.2 HSI_VALUE	6
3.5.2.3 MSI_VALUE	6
3.6 STM32L4xx_System_Private_Macros	6
3.7 STM32L4xx_System_Private_Variables	6
3.7.1 Detailed Description	6
3.7.2 Variable Documentation	6
3.7.2.1 MSIRangeTable	7
3.8 STM32L4xx_System_Private_FunctionPrototypes	7
3.9 STM32L4xx_System_Private_Functions	7
3.9.1 Detailed Description	7
3.9.2 Function Documentation	7
3.9.2.1 SystemCoreClockUpdate()	7
3.9.2.2 SystemInit()	8
4 File Documentation	9
4.1 Core/Src/adc.c File Reference	9
4.1.1 Detailed Description	9
4.1.2 Function Documentation	10
4.1.2.1 HAL_ADC_MspDeInit()	10
4.1.2.2 HAL_ADC_MspInit()	10
4.1.2.3 MX_ADC1_Init()	10
4.2 Core/Src/display.c File Reference	10
4.2.1 Detailed Description	11
4.2.2 Function Documentation	11
4.2.2.1 display_send_instruction()	11
4.2.2.2 display_transmit()	12

4.2.2.3 display_write()	12
4.2.2.4 display_write_row()	12
4.2.2.5 set_row()	13
4.2.2.6 split_byte()	13
4.3 Core/Src/gpio.c File Reference	13
4.3.1 Detailed Description	14
4.3.2 Function Documentation	14
4.3.2.1 MX_GPIO_Init()	14
4.4 Core/Src/main.c File Reference	14
4.4.1 Detailed Description	15
4.4.2 Function Documentation	15
4.4.2.1 Error_Handler()	15
4.4.2.2 HAL_TIM_PeriodElapsedCallback()	16
4.4.2.3 HAL_UART_RxCpltCallback()	16
4.4.2.4 HAL_UART_TxCpltCallback()	17
4.4.2.5 main()	17
4.4.2.6 MX_FREERTOS_Init()	17
4.4.2.7 SystemClock_Config()	18
4.5 Core/Src/potentiometer.c File Reference	18
4.5.1 Detailed Description	18
4.5.2 Function Documentation	18
4.5.2.1 get_potentiometer_value()	19
4.6 Core/Src/rtc.c File Reference	19
4.6.1 Detailed Description	19
4.6.2 Function Documentation	19
4.6.2.1 HAL_RTC_MspInit()	20
4.6.2.2 MX_RTC_Init()	20
4.7 Core/Src/spi.c File Reference	20
4.7.1 Detailed Description	20
4.7.2 Function Documentation	21
4.7.2.1 HAL_SPI_MspDeInit()	21
4.7.2.2 HAL_SPI_MspInit()	21
4.8 Core/Src/stm32l4xx_hal_msp.c File Reference	21
4.8.1 Detailed Description	21
4.8.2 Function Documentation	22
4.8.2.1 HAL_MspInit()	22
4.9 Core/Src/stm32l4xx_hal_timebase_tim.c File Reference	22
4.9.1 Detailed Description	22
4.9.2 Function Documentation	23
4.9.2.1 HAL_InitTick()	23
4.9.2.2 HAL_ResumeTick()	23
4.9.2.3 HAL_SuspendTick()	24

4.10 Core/Src/stm32l4xx_it.c File Reference	24
4.10.1 Detailed Description	25
4.11 Core/Src/syscalls.c File Reference	25
4.11.1 Detailed Description	26
4.12 Core/Src/systemem.c File Reference	26
4.12.1 Detailed Description	27
4.12.2 Function Documentation	27
4.12.2.1 _sbrk()	27
4.13 Core/Src/system_stm32l4xx.c File Reference	28
4.13.1 Detailed Description	28
4.13.2 This file configures the system clock as follows:	29
4.13.2.1 System Clock source MSI	29
4.13.2.2 SYSCLK(Hz) 4000000	29
4.13.2.3 HCLK(Hz) 4000000	29
4.13.2.4 AHB Prescaler 1	29
4.13.2.5 APB1 Prescaler 1	29
4.13.2.6 APB2 Prescaler 1	29
4.13.2.7 PLL_M 1	29
4.13.2.8 PLL_N 8	29
4.13.2.9 PLL_P 7	29
4.13.2.10 PLL_Q 2	29
4.13.2.11 PLL_R 2	29
4.13.2.12 PLLSAI1_P NA	29
4.13.2.13 PLLSAI1_Q NA	29
4.13.2.14 PLLSAI1_R NA	29
4.13.2.15 PLLSAI2_P NA	29
4.13.2.16 PLLSAI2_Q NA	29
4.13.2.17 PLLSAI2_R NA	29
4.13.2.18 SDIO and RNG clock 	29
4.14 Core/Src/tim.c File Reference	29
4.14.1 Detailed Description	30
4.14.2 Function Documentation	30
4.14.2.1 HAL_TIM_MspPostInit()	30
4.15 Core/Src/uart.c File Reference	30
4.15.1 Detailed Description	30
4.15.2 Function Documentation	31
4.15.2.1 uart_get_clock_input()	31
4.15.2.2 uart_println()	31
4.15.2.3 uart_printnum()	31
4.15.2.4 uart_receive()	31
4.15.2.5 uart_send()	31
4.16 Core/Src/usart.c File Reference	32

4.16.1 Detailed Description	32
4.16.2 Function Documentation	32
4.16.2.1 HAL_UART_MspDeInit()	32
4.16.2.2 HAL_UART_MspInit()	32
Index	33

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

CMSIS	5
Stm32l4xx_system	5
STM32L4xx_System_Private_Includes	5
STM32L4xx_System_Private_TypesDefinitions	5
STM32L4xx_System_Private_Defines	5
STM32L4xx_System_Private_Macros	6
STM32L4xx_System_Private_Variables	6
STM32L4xx_System_Private_FunctionPrototypes	7
STM32L4xx_System_Private_Functions	7

Chapter 2

File Index

2.1 File List

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

Core/Src/ adc.c	
This file provides code for the configuration of the ADC instances	9
Core/Src/ display.c	
This file provides code for initialising and communicating with the display module	10
Core/Src/ gpio.c	
This file provides code for the configuration of all used GPIO pins	13
Core/Src/ main.c	
: Main program body	14
Core/Src/ potentiometer.c	
This file provides code for reading the potentiometer value	18
Core/Src/ rtc.c	
This file provides code for the configuration of the RTC instances	19
Core/Src/ spi.c	
This file provides code for the configuration of the SPI instances	20
Core/Src/ stm32l4xx_hal_msp.c	
This file provides code for the MSP Initialization and de-Initialization codes	21
Core/Src/ stm32l4xx_hal_timebase_tim.c	
HAL time base based on the hardware TIM	22
Core/Src/ stm32l4xx_it.c	
Interrupt Service Routines	24
Core/Src/ syscalls.c	
STM32CubeIDE Minimal System calls file	25
Core/Src/ systemem.c	
STM32CubeIDE System Memory calls file	26
Core/Src/ system_stm32l4xx.c	
CMSIS Cortex-M4 Device Peripheral Access Layer System Source File	28
Core/Src/ tim.c	
This file provides code for the configuration of the TIM instances	29
Core/Src/ uart.c	
This file contains functions for communicating via UART	30
Core/Src/ usart.c	
This file provides code for the configuration of the USART instances	32

Chapter 3

Module Documentation

3.1 CMSIS

Modules

- [Stm32l4xx_system](#)

3.1.1 Detailed Description

3.2 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](#)

3.2.1 Detailed Description

3.3 STM32L4xx_System_Private_Includes

3.4 STM32L4xx_System_Private_TypesDefinitions

3.5 STM32L4xx_System_Private_Defines

Macros

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

3.5.1 Detailed Description

3.5.2 Macro Definition Documentation

3.5.2.1 HSE_VALUE

```
#define HSE_VALUE 8000000U
```

Value of the External oscillator in Hz

3.5.2.2 HSI_VALUE

```
#define HSI_VALUE 16000000U
```

Value of the Internal oscillator in Hz

3.5.2.3 MSI_VALUE

```
#define MSI_VALUE 4000000U
```

Value of the Internal oscillator in Hz

3.6 STM32L4xx_System_Private_Macros

3.7 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]

3.7.1 Detailed Description

3.7.2 Variable Documentation

3.7.2.1 MSIRangeTable

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

Initial value:

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

3.8 STM32L4xx_System_Private_FunctionPrototypes

3.9 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.

3.9.1 Detailed Description

3.9.2 Function Documentation

3.9.2.1 SystemCoreClockUpdate()

```
void SystemCoreClockUpdate (
    void )
```

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

Note

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

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

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

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

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

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

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

Return values

<i>None</i>	
-------------	--

3.9.2.2 SystemInit()

```
void SystemInit (  
                void )
```

Setup the microcontroller system.

Return values

<i>None</i>	
-------------	--

Chapter 4

File Documentation

4.1 Core/Src/adc.c File Reference

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

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

Functions

- void [MX_ADC1_Init](#) (void)
- void [HAL_ADC_MspInit](#) (ADC_HandleTypeDef *adcHandle)
- void [HAL_ADC_MspDeInit](#) (ADC_HandleTypeDef *adcHandle)

Variables

- ADC_HandleTypeDef **hadc1**

4.1.1 Detailed Description

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

Attention

© Copyright (c) 2021 STMicroelectronics. All rights reserved.

This software component is licensed by ST under Ultimate Liberty license SLA0044, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: www.st.com/SLA0044

4.1.2 Function Documentation

4.1.2.1 HAL_ADC_MspDeInit()

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

ADC1 GPIO Configuration PB1 ----> ADC1_IN16

4.1.2.2 HAL_ADC_MspInit()

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

Initializes the peripherals clock

ADC1 GPIO Configuration PB1 ----> ADC1_IN16

4.1.2.3 MX_ADC1_Init()

```
void MX_ADC1_Init (
    void )
```

Common config

Configure the ADC multi-mode

Configure Regular Channel

4.2 Core/Src/display.c File Reference

This file provides code for initialising and communicating with the display module.

```
#include "main.h"
#include "spi.h"
#include "error.h"
#include "red.h"
```


Functions

- void `hardware_reset` ()
Perform a hardware reset on the display Resets the display by writing to the displays hardware reset pin.
- void `test_backlight` ()
Test all backlight colors Run through each color of the display to see that they are lighting up.
- void `set_backlight` (uint8_t color, GPIO_PinState state)
Set a backlight color.
- void `split_byte` (uint8_t byte, uint8_t *buffer)
Split a byte to send to the display.
- int `display_transmit` (uint8_t startbyte, uint8_t *bytes, uint16_t length)
Send the display data or instructions.
- int `display_send_instruction` (uint8_t *instructions, uint16_t length)
Send instruction bytes via spi to the display.
- int `display_write` (char *characters, uint16_t length)
Write characters to the display where the cursor currently are.
- int `set_row` (uint8_t row)
Set the cursor on the display.
- int `display_write_row` (char *characters, uint16_t length, uint8_t row)
Write text to a specific row on the display.
- int `clear_display` ()
Clears the display.
- void `init_display` ()
Initialise the display.

Variables

- GPIO_TypeDef * **ports** [] = {Disp_White_GPIO_Port, Disp_Green_GPIO_Port}
- uint16_t **pins** [] = {Disp_White_Pin, Disp_Green_Pin}
- uint8_t **rows** [] = {0b10000000, 0b10100000, 0b11000000, 0b11100000}

4.2.1 Detailed Description

This file provides code for initialising and communicating with the display module.

Author

William Asp

4.2.2 Function Documentation

4.2.2.1 display_send_instruction()

```
int display_send_instruction (
    uint8_t * instructions,
    uint16_t length )
```

Send instruction bytes via spi to the display.

Parameters

in	<i>instructions</i>	A pointer to the instructions to send to the display
in	<i>length</i>	The number of instructions

4.2.2.2 display_transmit()

```
int display_transmit (
    uint8_t startbyte,
    uint8_t * bytes,
    uint16_t length )
```

Send the display data or instructions.

Parameters

in	<i>startbyte</i>	The byte setting that initiates the transmit
in	<i>bytes</i>	The bytes that will be sent to the display
in	<i>length</i>	The number of bytes to send

4.2.2.3 display_write()

```
int display_write (
    char * characters,
    uint16_t length )
```

Write characters to the display where the cursor currently are.

Parameters

<i>characters</i>	The characters to write
<i>length</i>	The number of characters

4.2.2.4 display_write_row()

```
int display_write_row (
    char * characters,
    uint16_t length,
    uint8_t row )
```

Write text to a specific row on the display.

Parameters

in	<i>characters</i>	The characters to write
in	<i>length</i>	The number of characters
in	<i>row</i>	The row to write to

4.2.2.5 set_row()

```
int set_row (
    uint8_t row )
```

Set the cursor on the display.

Parameters

in	<i>row</i>	The row to write to
----	------------	---------------------

4.2.2.6 split_byte()

```
void split_byte (
    uint8_t byte,
    uint8_t * buffer )
```

Split a byte to send to the display.

Parameters

in	<i>byte</i>	The byte to split into two
out	<i>buffer</i>	Where to place the two new bytes

4.3 Core/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)

4.3.1 Detailed Description

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

Attention

© Copyright (c) 2021 STMicroelectronics. All rights reserved.

This software component is licensed by ST under Ultimate Liberty license SLA0044, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: www.st.com/SLA0044

4.3.2 Function Documentation

4.3.2.1 MX_GPIO_Init()

```
void MX_GPIO_Init (
    void )
```

Configure pins as Analog Input Output EVENT_OUT EXTI

4.4 Core/Src/main.c File Reference

: Main program body

```
#include "main.h"
#include "cmsis_os.h"
#include "adc.h"
#include "rtc.h"
#include "spi.h"
#include "tim.h"
#include "gpio.h"
#include "string.h"
#include "usart.h"
#include "stdio.h"
#include "display.h"
#include "error.h"
#include "uart.h"
#include "clock.h"
#include "red.h"
```

Functions

- void [SystemClock_Config](#) (void)
System Clock Configuration.
- void [MX_FREERTOS_Init](#) (void)
FreeRTOS initialization.
- void [HAL_UART_RxCpltCallback](#) (UART_HandleTypeDef *UartHandle)
Rx Transfer completed callback.
- void [HAL_UART_TxCpltCallback](#) (UART_HandleTypeDef *UartHandle)
Tx Transfer completed callback.
- 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.

Variables

- ITStatus [uartReady](#) = RESET

4.4.1 Detailed Description

: Main program body

Attention

© Copyright (c) 2021 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

4.4.2 Function Documentation

4.4.2.1 Error_Handler()

```
void Error_Handler (  
    void )
```

This function is executed in case of error occurrence.

Return values

<i>None</i>	
-------------	--

4.4.2.2 HAL_TIM_PeriodElapsedCallback()

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

Period elapsed callback in non blocking mode.

Note

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

Parameters

<i>htim</i>	: TIM handle
-------------	--------------

Return values

<i>None</i>	
-------------	--

4.4.2.3 HAL_UART_RxCpltCallback()

```
void HAL_UART_RxCpltCallback (
    UART_HandleTypeDef * UartHandle )
```

Rx Transfer completed callback.

Parameters

<i>UartHandle</i>	UART handle
-------------------	-------------

Note

This example shows a simple way to report end of IT Rx transfer, and you can add your own implementation.

Return values

<i>None</i>	
-------------	--

4.4.2.4 HAL_UART_TxCpltCallback()

```
void HAL_UART_TxCpltCallback (
    UART_HandleTypeDef * UartHandle )
```

Tx Transfer completed callback.

Parameters

<i>UartHandle</i>	UART handle.
-------------------	--------------

Note

This example shows a simple way to report end of IT Tx transfer, and you can add your own implementation.

Return values

<i>None</i>	
-------------	--

4.4.2.5 main()

```
int main (
    void )
```

The application entry point.

Return values

<i>int</i>	
------------	--

4.4.2.6 MX_FREERTOS_Init()

```
void MX_FREERTOS_Init (
    void )
```

FreeRTOS initialization.

Parameters

<i>None</i>	
-------------	--

Return values

None	
------	--

4.4.2.7 SystemClock_Config()

```
void SystemClock_Config (  
    void )
```

System Clock Configuration.

Return values

None	
------	--

Configure the main internal regulator output voltage

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

Initializes the CPU, AHB and APB buses clocks

4.5 Core/Src/potentiometer.c File Reference

This file provides code for reading the potentiometer value.

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

Functions

- uint32_t [get_potentiometer_value](#) ()
Read the potentiometer value.

4.5.1 Detailed Description

This file provides code for reading the potentiometer value.

Author

William Asp

4.5.2 Function Documentation

4.5.2.1 get_potentiometer_value()

```
uint32_t get_potentiometer_value ( )
```

Read the potentiometer value.

Returns

The value of the potentiometer

4.6 Core/Src/rtc.c File Reference

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

```
#include "rtc.h"
```

Functions

- void [MX_RTC_Init](#) (void)
- void [HAL_RTC_MspInit](#) (RTC_HandleTypeDef *rtcHandle)
- void [HAL_RTC_MspDeInit](#) (RTC_HandleTypeDef *rtcHandle)

Variables

- RTC_HandleTypeDef [hrtc](#)

4.6.1 Detailed Description

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

Attention

© Copyright (c) 2021 STMicroelectronics. All rights reserved.

This software component is licensed by ST under Ultimate Liberty license SLA0044, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: www.st.com/SLA0044

4.6.2 Function Documentation

4.6.2.1 HAL_RTC_MspInit()

```
void HAL_RTC_MspInit (
    RTC_HandleTypeDef * rtcHandle )
```

Initializes the peripherals clock

4.6.2.2 MX_RTC_Init()

```
void MX_RTC_Init (
    void )
```

Initialize RTC Only

4.7 Core/Src/spi.c File Reference

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

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

Functions

- void **MX_SPI2_Init** (void)
- void [HAL_SPI_MspInit](#) (SPI_HandleTypeDef *spiHandle)
- void [HAL_SPI_MspDeInit](#) (SPI_HandleTypeDef *spiHandle)

Variables

- SPI_HandleTypeDef **hspi2**

4.7.1 Detailed Description

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

Attention

© Copyright (c) 2021 STMicroelectronics. All rights reserved.

This software component is licensed by ST under Ultimate Liberty license SLA0044, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: www.st.com/SLA0044

4.7.2 Function Documentation

4.7.2.1 HAL_SPI_MspDeInit()

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

SPI2 GPIO Configuration PC3 ----> SPI2_MOSI PB10 ----> SPI2_SCK PB12 ----> SPI2_NSS

4.7.2.2 HAL_SPI_MspInit()

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

SPI2 GPIO Configuration PC3 ----> SPI2_MOSI PB10 ----> SPI2_SCK PB12 ----> SPI2_NSS

4.8 Core/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)

4.8.1 Detailed Description

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

Attention

© Copyright (c) 2021 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

4.8.2 Function Documentation

4.8.2.1 HAL_MspInit()

```
void HAL_MspInit (
    void )
```

Initializes the Global MSP.

4.9 Core/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**

4.9.1 Detailed Description

HAL time base based on the hardware TIM.

Attention

© Copyright (c) 2021 STMicroelectronics. All rights reserved.

This software component is licensed by ST under Ultimate Liberty license SLA0044, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: www.st.com/SLA0044

4.9.2 Function Documentation

4.9.2.1 HAL_InitTick()

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

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

Note

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

Parameters

<i>TickPriority</i>	Tick interrupt priority.
---------------------	--------------------------

Return values

<i>HAL</i>	status
------------	--------

4.9.2.2 HAL_ResumeTick()

```
void HAL_ResumeTick (
    void )
```

Resume Tick increment.

Note

Enable the tick increment by Enabling TIM1 update interrupt.

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

4.9.2.3 HAL_SuspendTick()

```
void HAL_SuspendTick (
    void )
```

Suspend Tick increment.

Note

Disable the tick increment by disabling TIM1 update interrupt.

Parameters

None	
------	--

Return values

None	
------	--

4.10 Core/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.
- void [UART5_IRQHandler](#) (void)
This function handles UART5 global interrupt.

Variables

- UART_HandleTypeDef **huart5**
- TIM_HandleTypeDef **htim1**

4.10.1 Detailed Description

Interrupt Service Routines.

Attention

© Copyright (c) 2021 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

4.11 Core/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`

4.11.1 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

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

4.12 Core/Src/system.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

4.12.1 Detailed Description

STM32CubeIDE System Memory calls file.

Author

Generated by STM32CubeIDE

For more information about which C functions
need which of these lowlevel functions
please consult the newlib libc manual

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

4.12.2 Function Documentation

4.12.2.1 `_sbrk()`

```
void* _sbrk (
    ptrdiff_t incr )
```

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

```
* #####
* # .data # .bss #          newlib heap          #          MSP stack          #
* #          #          #          #          # Reserved by _Min_Stack_Size #
* #####
* ^-- RAM start          ^-- _end          _estack, RAM end --^
*
```

This implementation starts allocating at the '`_end`' linker symbol The '`_Min_Stack_Size`' linker symbol reserves a memory for the MSP stack The implementation considers '`_estack`' linker symbol to be RAM end NOTE: If the MSP stack, at any point during execution, grows larger than the reserved size, please increase the '`_Min_Stack_Size`'.

Parameters

<i>incr</i>	Memory size
-------------	-------------

Returns

Pointer to allocated memory

4.13 Core/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]`

4.13.1 Detailed Description

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

Author

MCD Application Team

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

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

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

4.13.2 This file configures the system clock as follows:

4.13.2.1 System Clock source | MSI

4.13.2.2 SYCLK(Hz) | 4000000

4.13.2.3 HCLK(Hz) | 4000000

4.13.2.4 AHB Prescaler | 1

4.13.2.5 APB1 Prescaler | 1

4.13.2.6 APB2 Prescaler | 1

4.13.2.7 PLL_M | 1

4.13.2.8 PLL_N | 8

4.13.2.9 PLL_P | 7

4.13.2.10 PLL_Q | 2

4.13.2.11 PLL_R | 2

4.13.2.12 PLLSAI1_P | NA

4.13.2.13 PLLSAI1_Q | NA

4.13.2.14 PLLSAI1_R | NA

4.13.2.15 PLLSAI2_P | NA

4.13.2.16 PLLSAI2_Q | NA

4.13.2.17 PLLSAI2_R | NA

Require 48MHz for USB OTG FS, | Disabled

4.13.2.18 SDIO and RNG clock |

=====

Attention

Copyright (c) 2017 STMicroelectronics. All rights reserved.

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.

4.14 Core/Src/tim.c File Reference

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

```
#include "tim.h"
```

Functions

- void **MX_TIM3_Init** (void)
- void **HAL_TIM_Base_MspInit** (TIM_HandleTypeDef *tim_baseHandle)
- void **HAL_TIM_MspPostInit** (TIM_HandleTypeDef *timHandle)
- void **HAL_TIM_Base_MspDeInit** (TIM_HandleTypeDef *tim_baseHandle)

Variables

- TIM_HandleTypeDef **htim3**

4.14.1 Detailed Description

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

Attention

© Copyright (c) 2021 STMicroelectronics. All rights reserved.

This software component is licensed by ST under Ultimate Liberty license SLA0044, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: www.st.com/SLA0044

4.14.2 Function Documentation

4.14.2.1 HAL_TIM_MspPostInit()

```
void HAL_TIM_MspPostInit (
    TIM_HandleTypeDef * timHandle )
TIM3 GPIO Configuration PC7 ----> TIM3_CH2
```

4.15 Core/Src/uart.c File Reference

This file contains functions for communicating via UART.

```
#include "main.h"
#include "usart.h"
#include "string.h"
#include "stdio.h"
```

Macros

- `#define TIMEOUT 0xFFFFFFFF`

Functions

- int `uart_send` (char *buffer, uint16_t length)
Send a string over uart.
- int `uart_receive` (char *buffer, uint16_t length)
Recieve a string over uart.
- int `uart_printf` (char *string)
send a string line to uart
- int `uart_printnum` (uint32_t num)
Print a number over uart.
- void `uart_get_clock_input` (char *buffer)
Let user input the time.

4.15.1 Detailed Description

This file contains functions for communicating via UART.

Author

William Asp

4.15.2 Function Documentation

4.15.2.1 `uart_get_clock_input()`

```
void uart_get_clock_input (
    char * buffer )
```

Let user input the time.

Parameters

out	<i>buffer</i>	The buffer to write to
-----	---------------	------------------------

4.15.2.2 `uart_println()`

```
int uart_println (
    char * string )
```

send a string line to uart

Parameters

in	<i>string</i>	The string to send
----	---------------	--------------------

4.15.2.3 `uart_printnum()`

```
int uart_printnum (
    uint32_t num )
```

Print a number over uart.

Parameters

in	<i>num</i>	The number to be printed over UART
----	------------	------------------------------------

4.15.2.4 `uart_receive()`

```
int uart_receive (
    char * buffer,
    uint16_t length )
```

Recieve a string over uart.

Parameters

out	<i>buffer</i>	The place to write the recieved string
in	<i>length</i>	The amount of data to read

4.15.2.5 `uart_send()`

```
int uart_send (
    char * buffer,
```

```
uint16_t length )
```

Send a string over uart.

Parameters

in	<i>message</i>	The character array to send
----	----------------	-----------------------------

Return values

<i>HAL</i>	status of uart transmission
------------	-----------------------------

4.16 Core/Src/usart.c File Reference

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

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

Functions

- void **MX_UART5_Init** (void)
- void [HAL_UART_MspInit](#) (UART_HandleTypeDef *uartHandle)
- void [HAL_UART_MspDeInit](#) (UART_HandleTypeDef *uartHandle)

Variables

- UART_HandleTypeDef **huart5**

4.16.1 Detailed Description

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

Attention

© Copyright (c) 2021 STMicroelectronics. All rights reserved.

This software component is licensed by ST under Ultimate Liberty license SLA0044, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: www.st.com/SLA0044

4.16.2 Function Documentation

4.16.2.1 HAL_UART_MspDeInit()

```
void HAL_UART_MspDeInit (
    UART_HandleTypeDef * uartHandle )
UART5 GPIO Configuration PC12 -----> UART5_TX PD2 -----> UART5_RX
```

4.16.2.2 HAL_UART_MspInit()

```
void HAL_UART_MspInit (
    UART_HandleTypeDef * uartHandle )
Initializes the peripherals clock
UART5 GPIO Configuration PC12 -----> UART5_TX PD2 -----> UART5_RX
```

Index

- [_sbrk](#)
 - [sysmem.c, 27](#)
- [adc.c](#)
 - [HAL_ADC_MspDeInit, 10](#)
 - [HAL_ADC_MspInit, 10](#)
 - [MX_ADC1_Init, 10](#)
- [CMSIS, 5](#)
- [Core/Src/adc.c, 9](#)
- [Core/Src/display.c, 10](#)
- [Core/Src/gpio.c, 13](#)
- [Core/Src/main.c, 14](#)
- [Core/Src/potentiometer.c, 18](#)
- [Core/Src/rtc.c, 19](#)
- [Core/Src/spi.c, 20](#)
- [Core/Src/stm32l4xx_hal_msp.c, 21](#)
- [Core/Src/stm32l4xx_hal_timebase_tim.c, 22](#)
- [Core/Src/stm32l4xx_it.c, 24](#)
- [Core/Src/syscalls.c, 25](#)
- [Core/Src/sysmem.c, 26](#)
- [Core/Src/system_stm32l4xx.c, 28](#)
- [Core/Src/tim.c, 29](#)
- [Core/Src/uart.c, 30](#)
- [Core/Src/usart.c, 32](#)

- [display.c](#)
 - [display_send_instruction, 11](#)
 - [display_transmit, 12](#)
 - [display_write, 12](#)
 - [display_write_row, 12](#)
 - [set_row, 13](#)
 - [split_byte, 13](#)
- [display_send_instruction](#)
 - [display.c, 11](#)
- [display_transmit](#)
 - [display.c, 12](#)
- [display_write](#)
 - [display.c, 12](#)
- [display_write_row](#)
 - [display.c, 12](#)

- [Error_Handler](#)
 - [main.c, 15](#)
- [get_potentiometer_value](#)
 - [potentiometer.c, 18](#)
- [gpio.c](#)
 - [MX_GPIO_Init, 14](#)
- [HAL_ADC_MspDeInit](#)
 - [adc.c, 10](#)
- [HAL_ADC_MspInit](#)
 - [adc.c, 10](#)
- [HAL_InitTick](#)
 - [stm32l4xx_hal_timebase_tim.c, 23](#)
- [HAL_MspInit](#)
 - [stm32l4xx_hal_msp.c, 22](#)
- [HAL_ResumeTick](#)
 - [stm32l4xx_hal_timebase_tim.c, 23](#)
- [HAL_RTC_MspInit](#)
 - [rtc.c, 19](#)
- [HAL_SPI_MspDeInit](#)
 - [spi.c, 21](#)
- [HAL_SPI_MspInit](#)
 - [spi.c, 21](#)
- [HAL_SuspendTick](#)
 - [stm32l4xx_hal_timebase_tim.c, 23](#)
- [HAL_TIM_MspPostInit](#)
 - [tim.c, 30](#)
- [HAL_TIM_PeriodElapsedCallback](#)
 - [main.c, 16](#)
- [HAL_UART_MspDeInit](#)
 - [usart.c, 32](#)
- [HAL_UART_MspInit](#)
 - [usart.c, 32](#)
- [HAL_UART_RxCpltCallback](#)
 - [main.c, 16](#)
- [HAL_UART_TxCpltCallback](#)
 - [main.c, 17](#)
- [HSE_VALUE](#)
 - [STM32L4xx_System_Private_Defines, 6](#)
- [HSI_VALUE](#)
 - [STM32L4xx_System_Private_Defines, 6](#)

- [main](#)
 - [main.c, 17](#)
- [main.c](#)
 - [Error_Handler, 15](#)
 - [HAL_TIM_PeriodElapsedCallback, 16](#)
 - [HAL_UART_RxCpltCallback, 16](#)
 - [HAL_UART_TxCpltCallback, 17](#)
 - [main, 17](#)
 - [MX_FREERTOS_Init, 17](#)
 - [SystemClock_Config, 18](#)
- [MSI_VALUE](#)
 - [STM32L4xx_System_Private_Defines, 6](#)
- [MSIRangeTable](#)
 - [STM32L4xx_System_Private_Variables, 6](#)
- [MX_ADC1_Init](#)
 - [adc.c, 10](#)

- MX_FREERTOS_Init
 - main.c, [17](#)
- MX_GPIO_Init
 - gpio.c, [14](#)
- MX_RTC_Init
 - rtc.c, [20](#)
- potentiometer.c
 - get_potentiometer_value, [18](#)
- rtc.c
 - HAL_RTC_MspInit, [19](#)
 - MX_RTC_Init, [20](#)
- set_row
 - display.c, [13](#)
- spi.c
 - HAL_SPI_MspDeInit, [21](#)
 - HAL_SPI_MspInit, [21](#)
- split_byte
 - display.c, [13](#)
- stm32l4xx_hal_msp.c
 - HAL_MspInit, [22](#)
- stm32l4xx_hal_timebase_tim.c
 - HAL_InitTick, [23](#)
 - HAL_ResumeTick, [23](#)
 - HAL_SuspendTick, [23](#)
- Stm32l4xx_system, [5](#)
- STM32L4xx_System_Private_Defines, [5](#)
 - HSE_VALUE, [6](#)
 - HSI_VALUE, [6](#)
 - MSI_VALUE, [6](#)
- STM32L4xx_System_Private_FunctionPrototypes, [7](#)
- STM32L4xx_System_Private_Functions, [7](#)
 - SystemCoreClockUpdate, [7](#)
 - SystemInit, [8](#)
- STM32L4xx_System_Private_Includes, [5](#)
- STM32L4xx_System_Private_Macros, [6](#)
- STM32L4xx_System_Private_TypesDefinitions, [5](#)
- STM32L4xx_System_Private_Variables, [6](#)
 - MSIRangeTable, [6](#)
- sysmem.c
 - _sbrk, [27](#)
- SystemClock_Config
 - main.c, [18](#)
- SystemCoreClockUpdate
 - STM32L4xx_System_Private_Functions, [7](#)
- SystemInit
 - STM32L4xx_System_Private_Functions, [8](#)
- tim.c
 - HAL_TIM_MspPostInit, [30](#)
- uart.c
 - uart_get_clock_input, [31](#)
 - uart_println, [31](#)
 - uart_prinnum, [31](#)
 - uart_receive, [31](#)
 - uart_send, [31](#)
- uart_get_clock_input
 - uart.c, [31](#)
- uart_println
 - uart.c, [31](#)
- uart_prinnum
 - uart.c, [31](#)
- uart_receive
 - uart.c, [31](#)
- uart_send
 - uart.c, [31](#)
- usart.c
 - HAL_UART_MspDeInit, [32](#)
 - HAL_UART_MspInit, [32](#)