ME 507 HW #4

Generated by Doxygen 1.10.0

Chapter 1

Topic Index

1.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Topic Documentation

3.1 CMSIS

Topics

- Stm32l4xx_system

3.1.1 Detailed Description

3.1.2 Stm32l4xx_system

Topics

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

3.1.2.1 Detailed Description

3.1.2.2 STM32L4xx_System_Private_Includes

3.1.2.3 STM32L4xx_System_Private_TypesDefinitions

3.1.2.4 STM32L4xx_System_Private_Defines

Macros

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

3.1.2.4.1 Detailed Description

3.1.2.4.2 Macro Definition Documentation

3.1.2.4.2.1 HSE_VALUE

```
#define HSE_VALUE 8000000U
```

Value of the External oscillator in Hz

3.1.2.4.2.2 HSI_VALUE

```
#define HSI_VALUE 16000000U
```

Value of the Internal oscillator in Hz

3.1.2.4.2.3 MSI_VALUE

```
#define MSI_VALUE 4000000U
```

Value of the Internal oscillator in Hz

3.1.2.5 STM32L4xx_System_Private_Macros

3.1.2.6 STM32L4xx_System_Private_Variables

Variables

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

3.1.2.6.1 Detailed Description

3.1.2.6.2 Variable Documentation

3.1.2.6.2.1 MSIRangeTable

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

Initial value:

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

3.1.2.7 STM32L4xx_System_Private_FunctionPrototypes

3.1.2.8 STM32L4xx_System_Private_Functions

Functions

- void SystemInit (void)

  Setup the microcontroller system.
- void SystemCoreClockUpdate (void)

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

3.1.2.8.1 Detailed Description

3.1.2.8.2 Function Documentation

3.1.2.8.2.1 SystemCoreClockUpdate()

```
void SystemCoreClockUpdate (
          void  )
```

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

Note

> Each time the core clock (HCLK) changes, this function must be called to update SystemCoreClock variable value. Otherwise, any configuration based on this variable will be incorrect.
>
> - The system frequency computed by this function is not the real frequency in the chip. It is calculated based on the predefined constant and the selected clock source:

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

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

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

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

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

Return values

None	

3.1.2.8.2.2 SystemInit()

```
void SystemInit (
            void  )
```

Setup the microcontroller system.

Return values

None	

Chapter 4

File Documentation

4.1 Core/Src/main.c File Reference

: Main program body

```
#include "main.h"
#include "string.h"
#include <stdio.h>
#include <stdlib.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.

Variables

- TIM_HandleTypeDef **htim2**
- UART_HandleTypeDef **huart2**
- int **PWM_DC1** = 1000
- int **PWM_DC2** = 1000
- int **PWM_DC3** = 1000
- int **PWM_DC4** = 1000
- int **char_flg** = 0
- char **char_in** = 0
- char **buffer** [25] = {0}
- int **buff_index** = 0
- char **command** [4] = {0}
- int **PWM_in** = 0
- char **string** [2] = {0}
- char **agw** [ ] = "\nUser has entered data this acknowlages the input\n\n\n\n"

4.1.1 Detailed Description

: Main program body

Attention

4.1.2 Function Documentation

4.1.2.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

4.1.2.2 main()

```
int main (
             void  )
```

The application entry point.

Return values

int	

4.1.2.3 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

4.2 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_TIM_MspPostInit (TIM_HandleTypeDef ∗htim)
- void HAL_MspInit (void)
- void HAL_TIM_PWM_MspInit (TIM_HandleTypeDef ∗htim_pwm)

  TIM_PWM MSP Initialization This function configures the hardware resources used in this example.
- void HAL_TIM_PWM_MspDeInit (TIM_HandleTypeDef ∗htim_pwm)

  TIM_PWM MSP De-Initialization This function freeze the hardware resources used in this example.
- void HAL_UART_MspInit (UART_HandleTypeDef ∗huart)

  UART MSP Initialization This function configures the hardware resources used in this example.
- void HAL_UART_MspDeInit (UART_HandleTypeDef ∗huart)

  UART MSP De-Initialization This function freeze the hardware resources used in this example.

4.2.1 Detailed Description

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

Attention

4.2.2 Function Documentation

4.2.2.1 HAL_MspInit()

```
void HAL_MspInit (
            void )
```

Initializes the Global MSP.

4.2.2.2 HAL_TIM_MspPostInit()

```
void HAL_TIM_MspPostInit (
            TIM_HandleTypeDef * htim )
```

TIM2 GPIO Configuration PA0 ------> TIM2_CH1 PA1 ------> TIM2_CH2 PB10 ------> TIM2_CH3 PB11 ------> TIM2_CH4

4.2.2.3 HAL_TIM_PWM_MspDeInit()

```
void HAL_TIM_PWM_MspDeInit (
            TIM_HandleTypeDef * htim_pwm )
```

TIM_PWM MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

htim_pwm	TIM_PWM handle pointer

Return values

None	

4.2.2.4 HAL_TIM_PWM_MspInit()

```
void HAL_TIM_PWM_MspInit (
            TIM_HandleTypeDef * htim_pwm )
```

TIM_PWM MSP Initialization This function configures the hardware resources used in this example.

Parameters

htim_pwm	TIM_PWM handle pointer

Return values

None	

4.2.2.5 HAL_UART_MspDeInit()

```
void HAL_UART_MspDeInit (
            UART_HandleTypeDef * huart )
```

UART MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

huart	UART handle pointer

Return values

None	

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

4.2.2.6 HAL_UART_MspInit()

```
void HAL_UART_MspInit (
            UART_HandleTypeDef * huart )
```

UART MSP Initialization This function configures the hardware resources used in this example.

Parameters

huart	UART handle pointer

Return values

None	

Initializes the peripherals clock

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

4.3 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 **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 **USART2_IRQHandler** (void)

  This function handles USART2 global interrupt.

Variables

- UART_HandleTypeDef **huart2**

4.3.1 Detailed Description

Interrupt Service Routines.

Attention

4.4 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

4.4.1 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) 2020-2024 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.

4.5 Core/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

4.5.1 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

4.5.2 Function Documentation

4.5.2.1 _sbrk()

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

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

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

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

Parameters

incr	Memory size

Returns

Pointer to allocated memory

4.6 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]

4.6.1 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.

4.6.2 This file configures the system clock as follows:

4.6.2.1 System Clock source | MSI

4.6.2.2 SYSCLK(Hz) | 4000000

4.6.2.3 HCLK(Hz) | 4000000

4.6.2.4 AHB Prescaler | 1

4.6.2.5 APB1 Prescaler | 1

4.6.2.6 APB2 Prescaler | 1

4.6.2.7 PLL_M | 1

4.6.2.8 PLL_N | 8

4.6.2.9 PLL_P | 7

4.6.2.10 PLL_Q | 2

4.6.2.11 PLL_R | 2

4.6.2.12 PLLSAI1_P | NA

4.6.2.13 PLLSAI1_Q | NA

4.6.2.14 PLLSAI1_R | NA

4.6.2.15 PLLSAI2_P | NA

4.6.2.16 PLLSAI2_Q | NA

4.6.2.17 PLLSAI2_R | NA

Require 48MHz for USB OTG FS, | Disabled

4.6.2.18 SDIO and RNG clock |

===

Attention

Index