ME507

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 files with brief descriptions:

Chapter 3

Topic Documentation

3.1 CMSIS

Topics

- Stm32f4xx_system

3.1.1 Detailed Description

3.1.2 Stm32f4xx_system

Topics

- STM32F4xx_System_Private_Includes
- STM32F4xx_System_Private_TypesDefinitions
- STM32F4xx_System_Private_Defines
- STM32F4xx_System_Private_Macros
- STM32F4xx_System_Private_Variables
- STM32F4xx_System_Private_FunctionPrototypes
- STM32F4xx_System_Private_Functions

3.1.2.1 Detailed Description

3.1.2.2 STM32F4xx_System_Private_Includes

Macros

- #define HSE_VALUE ((uint32_t)25000000)
- #define HSI_VALUE ((uint32_t)16000000)

3.1.2.2.1 Detailed Description

3.1.2.2.2 Macro Definition Documentation

3.1.2.2.2.1 HSE_VALUE

```
#define HSE_VALUE ((uint32_t)25000000)
```

Default value of the External oscillator in Hz

3.1.2.2.2.2 HSI_VALUE

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

Value of the Internal oscillator in Hz

3.1.2.3 STM32F4xx_System_Private_TypesDefinitions

3.1.2.4 STM32F4xx_System_Private_Defines

3.1.2.5 STM32F4xx_System_Private_Macros

3.1.2.6 STM32F4xx_System_Private_Variables

Variables

- uint32_t SystemCoreClock = 16000000
- const uint8_t AHBPrescTable [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t APBPrescTable [8] = {0, 0, 0, 0, 1, 2, 3, 4}

3.1.2.6.1 Detailed Description

3.1.2.6.2 Variable Documentation

3.1.2.6.2.1 AHBPrescTable

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

3.1.2.6.2.2 APBPrescTable

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

3.1.2.6.2.3 SystemCoreClock

```
uint32_t SystemCoreClock = 16000000
```

3.1.2.7 STM32F4xx_System_Private_FunctionPrototypes

3.1.2.8 STM32F4xx_System_Private_Functions

Functions

- void SystemInit (void)

  Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.
- 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 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 stm32f4xx_hal_conf.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 stm32f4xx_hal_conf.h file (its value depends on the application requirements), 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.

Parameters

None	

Return values

None	

3.1.2.8.2.2 SystemInit()

```
void SystemInit (
             void  )
```

Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.

Parameters

None	

Return values

None	

Chapter 4

File Documentation

4.1 Lab3/Core/Src/main.c File Reference

: Main program body

```
#include "main.h"
#include "motor_driver.h"
#include <stdio.h>
```

Macros

- #define TIMCLOCK 96000000
- #define TIMCLOCK2 96000000
- #define PRESCALAR 0

Functions

- void SystemClock_Config (void)

  System Clock Configuration.
- int main (void)

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

  This function is executed in case of error occurrence.

Variables

- int dc
- int dc2
- int test
- uint32_t IC_Val1 = 0
- uint32_t IC_Val2 = 0
- uint32_t Difference = 0
- int Is_First_Captured = 0
- int ifc = 0
- uint32_t IC_Val12 = 0
- uint32_t IC_Val22 = 0
- uint32_t Difference2 = 0
- uint32_t usWidth = 0
- uint32_t usWidth2 = 0
- motor_t mot1
- motor_t mot2
- int n
- char buffer [50]
- TIM_HandleTypeDef htim1
- TIM_HandleTypeDef htim2
- UART_HandleTypeDef huart1

4.1.1 Detailed Description

: Main program body

Attention

4.1.2 Macro Definition Documentation

4.1.2.1 PRESCALAR

```
#define PRESCALAR 0
```

4.1.2.2 TIMCLOCK

```
#define TIMCLOCK 96000000
```

4.1.2.3 TIMCLOCK2

```
#define TIMCLOCK2 96000000
```

4.1.3 Function Documentation

4.1.3.1 Error_Handler()

```
void Error_Handler (
            void  )
```

This function is executed in case of error occurrence.

Return values

None	

4.1.3.2 HAL_TIM_IC_CaptureCallback()

```
void HAL_TIM_IC_CaptureCallback (
             TIM_HandleTypeDef * htim )
```

4.1.3.3 main()

```
int main (
             void  )
```

The application entry point.

Return values

int	

4.1.3.4 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

4.1.4 Variable Documentation

4.1.4.1 buffer

```
char buffer[50]
```

4.1.4.2 dc

```
int dc
```

4.1.4.3 dc2

```
int dc2
```

4.1.4.4 Difference

```
uint32_t Difference = 0
```

4.1.4.5 Difference2

```
uint32_t Difference2 = 0
```

4.1.4.6 htim1

```
TIM_HandleTypeDef htim1
```

4.1.4.7 htim2

```
TIM_HandleTypeDef htim2
```

4.1.4.8 huart1

```
UART_HandleTypeDef huart1
```

4.1.4.9 IC_Val1

```
uint32_t IC_Val1 = 0
```

4.1.4.10 IC_Val12

```
uint32_t IC_Val12 = 0
```

4.1.4.11 IC_Val2

```
uint32_t IC_Val2 = 0
```

4.1.4.12 IC_Val22

```
uint32_t IC_Val22 = 0
```

4.1.4.13 ifc

```
int ifc = 0
```

4.1.4.14 Is_First_Captured

```
int Is_First_Captured = 0
```

4.1.4.15 mot1

```
motor_t mot1
```

Initial value:

```
= {.duty    = 0,
                  .channel = 1,
                  .timer = TIM1}
```

4.1.4.16 mot2

```
motor_t mot2
```

Initial value:

```
= {.duty    = 0,
                  .channel = 2,
                  .timer = TIM1}
```

4.1.4.17 n

```
int n
```

4.1.4.18 test

```
int test
```

4.1.4.19 usWidth

```
uint32_t usWidth = 0
```

4.1.4.20 usWidth2

```
uint32_t usWidth2 = 0
```

4.2 Lab3/Core/Src/motor_driver.c File Reference

```
#include "motor_driver.h"
```

Functions

- void set_duty (motor_t *p_mot, int32_t duty)
- void start_PWM (TIM_HandleTypeDef *htim)
- void stop_PWM (TIM_HandleTypeDef *htim)

4.2.1 Function Documentation

4.2.1.1 set_duty()

```
void set_duty (
            motor_t * p_mot,
            int32_t duty )
```

4.2.1.2 start_PWM()

```
void start_PWM (
            TIM_HandleTypeDef * htim )
```

4.2.1.3 stop_PWM()

```
void stop_PWM (
            TIM_HandleTypeDef * htim )
```

4.3 Lab3/Core/Src/stm32f4xx_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_IC_MspInit (TIM_HandleTypeDef *htim_ic)

  TIM_IC 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_TIM_IC_MspDeInit (TIM_HandleTypeDef *htim_ic)

  TIM_IC 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.

4.3.1 Detailed Description

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

Attention

4.3.2 Function Documentation

4.3.2.1 HAL_MspInit()

```
void HAL_MspInit (
             void  )
```

Initializes the Global MSP.

4.3.2.2 HAL_TIM_IC_MspDeInit()

```
void HAL_TIM_IC_MspDeInit (
             TIM_HandleTypeDef * htim_ic )
```

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

Parameters

htim_ic	TIM_IC handle pointer

Return values

None	

TIM2 GPIO Configuration PA0-WKUP ------> TIM2_CH1 PA1 ------> TIM2_CH2

4.3.2.3 HAL_TIM_IC_MspInit()

```
void HAL_TIM_IC_MspInit (
             TIM_HandleTypeDef * htim_ic )
```

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

Parameters

htim_ic	TIM_IC handle pointer

Return values

None	

TIM2 GPIO Configuration PA0-WKUP ------> TIM2_CH1 PA1 ------> TIM2_CH2

4.3.2.4 HAL_TIM_MspPostInit()

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

TIM1 GPIO Configuration PA8 ------> TIM1_CH1 PA9 ------> TIM1_CH2 PA10 ------> TIM1_CH3 PA11 ------> TIM1_CH4

4.3.2.5 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	

4.3.2.6 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	

4.3.2.7 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	

USART1 GPIO Configuration PA15 ------> USART1_TX PB7 ------> USART1_RX

4.3.2.8 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	

USART1 GPIO Configuration PA15 ------> USART1_TX PB7 ------> USART1_RX

4.4 Lab3/Core/Src/stm32f4xx_it.c File Reference

Interrupt Service Routines.

```
#include "main.h"
#include "stm32f4xx_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 TIM2_IRQHandler (void)

  This function handles TIM2 global interrupt.

Variables

- TIM_HandleTypeDef htim2

4.4.1 Detailed Description

Interrupt Service Routines.

Attention

4.4.2 Function Documentation

4.4.2.1 BusFault_Handler()

```
void BusFault_Handler (
             void  )
```

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

4.4.2.2 DebugMon_Handler()

```
void DebugMon_Handler (
             void  )
```

This function handles Debug monitor.

4.4.2.3 HardFault_Handler()

```
void HardFault_Handler (
             void  )
```

This function handles Hard fault interrupt.

4.4.2.4 MemManage_Handler()

```
void MemManage_Handler (
             void  )
```

This function handles Memory management fault.

4.4.2.5 NMI_Handler()

```
void NMI_Handler (
             void  )
```

This function handles Non maskable interrupt.

4.4.2.6 PendSV_Handler()

```
void PendSV_Handler (
             void  )
```

This function handles Pendable request for system service.

4.4.2.7 SVC_Handler()

```
void SVC_Handler (
             void  )
```

This function handles System service call via SWI instruction.

4.4.2.8 SysTick_Handler()

```
void SysTick_Handler (
             void  )
```

This function handles System tick timer.

4.4.2.9 TIM2_IRQHandler()

```
void TIM2_IRQHandler (
            void  )
```

This function handles TIM2 global interrupt.

4.4.2.10 UsageFault_Handler()

```
void UsageFault_Handler (
            void  )
```

This function handles Undefined instruction or illegal state.

4.4.3 Variable Documentation

4.4.3.1 htim2

```
TIM_HandleTypeDef htim2  [extern]
```

4.5 Lab3/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

4.5.1 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

4.5.2 Function Documentation

4.5.2.1 __attribute__()

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

4.5.2.2 __io_getchar()

```
int __io_getchar (
             void  )  [extern]
```

4.5.2.3 __io_putchar()

```
int __io_putchar (
             int ch )  [extern]
```

4.5.2.4 _close()

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

4.5.2.5 _execve()

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

4.5.2.6 _exit()

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

4.5.2.7 _fork()

```
int _fork (
             void  )
```

4.5.2.8 _fstat()

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

4.5.2.9 _getpid()

```
int _getpid (
             void  )
```

4.5.2.10 _isatty()

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

4.5.2.11 _kill()

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

4.5.2.12 _link()

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

4.5.2.13 _lseek()

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

4.5.2.14 _open()

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

4.5.2.15 _stat()

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

4.5.2.16 _times()

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

4.5.2.17 _unlink()

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

4.5.2.18 _wait()

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

4.5.2.19 initialise_monitor_handles()

```
void initialise_monitor_handles ( )
```

4.5.3 Variable Documentation

4.5.3.1 environ

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

4.6 Lab3/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

4.6.1 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

4.6.2 Function Documentation

4.6.2.1 _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

4.7 Lab3/Core/Src/system_stm32f4xx.c File Reference

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

```
#include "stm32f4xx.h"
```

Macros

- #define HSE_VALUE ((uint32_t)25000000)
- #define HSI_VALUE ((uint32_t)16000000)

Functions

- void SystemInit (void)

  Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.
- 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 = 16000000
- const uint8_t AHBPrescTable [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t APBPrescTable [8] = {0, 0, 0, 0, 1, 2, 3, 4}

4.7.1 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_stm32f4xx.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

Index