TP MCC - CHAPUIS & SUTRA

Generated by Doxygen 1.9.5

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 documented files with brief descriptions:

Chapter 3

Module Documentation

3.1 CMSIS

Modules

- **Stm32g4xx_system**

3.1.1 Detailed Description

3.2 Stm32g4xx_system

Modules

- **STM32G4xx_System_Private_Includes**
- **STM32G4xx_System_Private_TypesDefinitions**
- **STM32G4xx_System_Private_Defines**
- **STM32G4xx_System_Private_Macros**
- **STM32G4xx_System_Private_Variables**
- **STM32G4xx_System_Private_FunctionPrototypes**
- **STM32G4xx_System_Private_Functions**

3.2.1 Detailed Description

3.3 STM32G4xx_System_Private_Includes

Macros

- #define **HSE_VALUE** 24000000U
- #define **HSI_VALUE** 16000000U

3.3.1 Detailed Description

3.3.2 Macro Definition Documentation

3.3.2.1 HSE_VALUE

```
#define HSE_VALUE 24000000U
```

Value of the External oscillator in Hz

3.3.2.2 HSI_VALUE

```
#define HSI_VALUE 16000000U
```

Value of the Internal oscillator in Hz

3.4 STM32G4xx_System_Private_TypesDefinitions

3.5 STM32G4xx_System_Private_Defines

3.6 STM32G4xx_System_Private_Macros

3.7 STM32G4xx_System_Private_Variables

Variables

- uint32_t **SystemCoreClock** = **HSI_VALUE**
- 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

3.8 STM32G4xx_System_Private_FunctionPrototypes

3.9 STM32G4xx_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 HSI, SystemCoreClock will contain the **HSI_VALUE(**$**$**)** (p. 6)
- If SYSCLK source is HSE, SystemCoreClock will contain the **HSE_VALUE(**$***$**)** (p. 6)
- If SYSCLK source is PLL, SystemCoreClock will contain the **HSE_VALUE(**$***$**)** (p. 6) or **HSI_VALUE(**$*$**)** (p. 6) multiplied/divided by the PLL factors.

($**$) HSI_VALUE is a constant defined in stm32g4xx_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 stm32g4xx_hal.h file (default value 24 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.

Parameters

None	

Return values

None	

3.9.2.2 SystemInit()

```
void SystemInit (
```

```
            void  )
```

Setup the microcontroller system.

Parameters

None	

Return values

None	

Chapter 4

File Documentation

4.1 main.c File Reference

: Main program body

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

Functions

- void **SystemClock_Config** (void)

  System Clock Configuration.
- void **newcom** (void)

  This function makes the initialization for the PI controller.
- void **delete** (void)

  This function allows us the delete letters in the shell.
- void **new_carac** (void)

  This function allows us to write a new letter in the shell.
- void **get** (void)

  This function writes a message in the shell in the case the command is not found.
- void **set** (void)

  This function switch on/off the led if the user writes PA5.
- void **help** (void)

  This function displays all the usable functions for the user.
- void **pinout** (void)

  This function shows all the connections for the pinouts.
- void **start** (void)

  This function makes the initialization for the chopper, the PWM for the motor and the initialization for the encoder.
- void **stop** (void)

  This function makes the initialization for the PI controller.
- void **set_speed** (char ∗vit)

  This function allows the user to set the motor speed.
- void **read_enc** (void)

  This function read the encoder value.
- void **read_speed** (void)

This function read the value of the speed (measured by the encoder).

- void **set_current** (char *set_cur)

  This function makes the conversion for the current value for the motor and its enslavement.

- float **get_current** ()

  This function make the measure of the current for the current enslavement.

- void **data** (void)

  This function shows all the data needed from the motor for this project: current value, encoder value and speed value.

- void **HAL_TIM_PeriodElapsedCallback** (TIM_HandleTypeDef *htim)

  The function which generate the interruptions.

- int **main** (void)

  The application entry point.

- void **HAL_UART_RxCpltCallback** (UART_HandleTypeDef *huart)
- void **HAL_ADC_ConvCpltCallback** (ADC_HandleTypeDef *hadc)
- void **Error_Handler** (void)

  This function is executed in case of error occurrence.

Variables

- ADC_HandleTypeDef **hadc1**
- DMA_HandleTypeDef **hdma_adc1**
- TIM_HandleTypeDef **htim1**
- TIM_HandleTypeDef **htim3**
- TIM_HandleTypeDef **htim4**
- UART_HandleTypeDef **huart2**
- uint8_t **prompt** [ ] ="user@Nucleo-STM32G474>>"
- uint8_t **started** [ ]
- uint8_t **newline** [ ] ="\r\n"
- uint8_t **cmdNotFound** [ ] ="Command not found\r\n"
- uint32_t **uartRxReceived**
- uint8_t **uartRxBuffer** [UART_RX_BUFFER_SIZE]
- uint8_t **uartTxBuffer** [UART_TX_BUFFER_SIZE]
- uint8_t **powerOn** [ ] ="Allumage du moteur\r\n"
- uint8_t **powerOff** [ ] ="Extinction du moteur\r\n"
- uint32_t **adcBuffer** [ADC_BUFFER_SIZE]
- char **cmdBuffer** [CMD_BUFFER_SIZE]
- int **idx_cmd**
- char * **argv** [MAX_ARGS]
- int **argc** = 0
- char * **token**
- int **newCmdReady** = 0
- int **abs_speed** = 50
- int **adcDMAFlag** = 0
- int **it_tim3** = 0
- int **it_tim1** = 0
- int **enc** = 0
- int **speed** = 0
- int **inc** = 0
- float **targetcur**
- char **pi_buffer** [256]
- PIController **pi** = {PI_KP, PI_KI, PI_LIM_MIN, PI_LIM_MAX, PI_LIM_MIN_INT, PI_LIM_MAX_INT, SAMPLE_TIME_S}

4.1.1 Detailed Description

: Main program body

Author

: Clément Chapuis & Aurélien Sutra

Date

: Date du dernier commit&push : 03/01/2023

Attention

4.1.2 Function Documentation

4.1.2.1 data()

```
void data (
             void  )
```

This function shows all the data needed from the motor for this project: current value, encoder value and speed value.

Parameters

in	*None*	

Return values

None	

4.1.2.2 delete()

```
void delete (
             void  )
```

This function allows us the delete letters in the shell.

Parameters

in	*None*	

Return values

None	

4.1.2.3 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

4.1.2.4 get()

```
void get (
             void  )
```

This function writes a message in the shell in the case the command is not found.

Parameters

in	*None*	

Return values

None	

4.1.2.5 get_current()

```
float get_current ( )
```

This function make the measure of the current for the current enslavement.

Parameters

in	*None*	

Return values

None	

4.1.2.6 HAL_TIM_PeriodElapsedCallback()

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

The function which generate the interruptions.

Parameters

in	*htim*	timer adress

Return values

None	

4.1.2.7 help()

```
void help (
            void  )
```

This function displays all the usable functions for the user.

Parameters

in	*None*	

Return values

None	

4.1.2.8 main()

```
int main (
```

```
            void  )
```

The application entry point.

Return values

int	

4.1.2.9 new_carac()

```
void new_carac (
            void  )
```

This function allows us to write a new letter in the shell.

Parameters

in	*None*	

Return values

None	

4.1.2.10 newcom()

```
void newcom (
            void  )
```

This function makes the initialization for the PI controller.

Parameters

in	*PIController*	The structure for all the PI parameters

Return values

None	

4.1.2.11 pinout()

```
void pinout (
            void  )
```

This function shows all the connections for the pinouts.

Parameters

in	*None*	

Return values

None	

4.1.2.12 read_enc()

```
void read_enc (
            void  )
```

This function read the encoder value.

Parameters

in	*None*	

Return values

None	

4.1.2.13 read_speed()

```
void read_speed (
            void  )
```

This function read the value of the speed (measured by the encoder).

Parameters

in	*None*	

Return values

None	

4.1.2.14 set()

```
void set (
            void  )
```

This function switch on/off the led if the user writes PA5.

Parameters

in	*None*	

Return values

None	

4.1.2.15 set_current()

```
void set_current (
            char * set_cur )
```

This function makes the conversion for the current value for the motor and its enslavement.

Parameters

in	*set_cur*	The value of current the user wants for the motor enslavement

Return values

None	

4.1.2.16 set_speed()

```
void set_speed (
            char * vit )
```

This function allows the user to set the motor speed.

Parameters

in	*vit*	The speed wanted by the user (0-49 in one direction, 51-100 in the other, 50 the motor doesn't turn)

Return values

None	

4.1.2.17 start()

```
void start (
            void  )
```

This function makes the initialization for the chopper, the PWM for the motor and the initialization for the encoder.

Parameters

in	*None*	

Return values

None	

4.1.2.18 stop()

```
void stop (
            void  )
```

This function makes the initialization for the PI controller.

Parameters

in	*None*	

Return values

None	

4.1.2.19 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

4.1.3 Variable Documentation

4.1.3.1 started

```
uint8_t started[]
```

Initial value:

```
=
        "\r\n*-----------------------------*"
        "\r\n| Welcome on Nucleo-STM32G474 |"
        "\r\n*-----------------------------*"
        "\r\n"
```

4.2 stm32g4xx_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_ADC_MspInit** (ADC_HandleTypeDef *hadc)

  ADC MSP Initialization This function configures the hardware resources used in this example.
- void **HAL_ADC_MspDeInit** (ADC_HandleTypeDef *hadc)

  ADC MSP De-Initialization This function freeze the hardware resources used in this example.
- void **HAL_TIM_Base_MspInit** (TIM_HandleTypeDef *htim_base)

  TIM_Base MSP Initialization This function configures the hardware resources used in this example.
- void **HAL_TIM_Encoder_MspInit** (TIM_HandleTypeDef *htim_encoder)

  TIM_Encoder MSP Initialization This function configures the hardware resources used in this example.
- void **HAL_TIM_Base_MspDeInit** (TIM_HandleTypeDef *htim_base)

  TIM_Base MSP De-Initialization This function freeze the hardware resources used in this example.
- void **HAL_TIM_Encoder_MspDeInit** (TIM_HandleTypeDef *htim_encoder)

  TIM_Encoder 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.

Variables

- DMA_HandleTypeDef **hdma_adc1**

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_ADC_MspDeInit()

```
void HAL_ADC_MspDeInit (
            ADC_HandleTypeDef * hadc )
```

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

Parameters

hadc	ADC handle pointer

Return values

None	

ADC1 GPIO Configuration PA0 ---—> ADC1_IN1

4.2.2.2 HAL_ADC_MspInit()

```
void HAL_ADC_MspInit (
            ADC_HandleTypeDef * hadc )
```

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

Parameters

hadc	ADC handle pointer

Return values

None	

Initializes the peripherals clocks

ADC1 GPIO Configuration PA0 ---—> ADC1_IN1

4.2.2.3 HAL_MspInit()

```
void HAL_MspInit (
             void  )
```

Initializes the Global MSP. Disable the internal Pull-Up in Dead Battery pins of UCPD peripheral

4.2.2.4 HAL_TIM_Base_MspDeInit()

```
void HAL_TIM_Base_MspDeInit (
             TIM_HandleTypeDef * htim_base )
```

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

Parameters

htim_base	TIM_Base handle pointer

Return values

None	

4.2.2.5 HAL_TIM_Base_MspInit()

```
void HAL_TIM_Base_MspInit (
             TIM_HandleTypeDef * htim_base )
```

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

Parameters

htim_base	TIM_Base handle pointer

Return values

None	

4.2.2.6 HAL_TIM_Encoder_MspDeInit()

```
void HAL_TIM_Encoder_MspDeInit (
            TIM_HandleTypeDef * htim_encoder )
```

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

Parameters

htim_encoder	TIM_Encoder handle pointer

Return values

None	

TIM4 GPIO Configuration PB6 ------> TIM4_CH1 PB7 ------> TIM4_CH2

4.2.2.7 HAL_TIM_Encoder_MspInit()

```
void HAL_TIM_Encoder_MspInit (
            TIM_HandleTypeDef * htim_encoder )
```

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

Parameters

htim_encoder	TIM_Encoder handle pointer

Return values

None	

TIM4 GPIO Configuration PB6 ------> TIM4_CH1 PB7 ------> TIM4_CH2

4.2.2.8 HAL_TIM_MspPostInit()

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

TIM1 GPIO Configuration PA8 ------> TIM1_CH1 PA9 ------> TIM1_CH2 PA11 ------> TIM1_CH1N PA12 ------> TIM1_CH2N

4.2.2.9 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

4.2.2.10 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 clocks

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

4.3 stm32g4xx_it.c File Reference

Interrupt Service Routines.

```
#include "main.h"
#include "stm32g4xx_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 **EXTI1_IRQHandler** (void)

  This function handles EXTI line1 interrupt.
- void **DMA1_Channel1_IRQHandler** (void)

  This function handles DMA1 channel1 global interrupt.
- void **TIM1_UP_TIM16_IRQHandler** (void)

  This function handles TIM1 update interrupt and TIM16 global interrupt.
- void **TIM3_IRQHandler** (void)

  This function handles TIM3 global interrupt.
- void **USART2_IRQHandler** (void)

  This function handles USART2 global interrupt / USART2 wake-up interrupt through EXTI line 26.
- void **EXTI15_10_IRQHandler** (void)

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

Variables

- DMA_HandleTypeDef **hdma_adc1**
- TIM_HandleTypeDef **htim1**
- TIM_HandleTypeDef **htim3**
- UART_HandleTypeDef **huart2**

4.3.1 Detailed Description

Interrupt Service Routines.

Attention

4.4 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

4.5 sysmem.c File Reference

STM32CubeIDE System Memory calls file.

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

Functions

- void * **_sbrk** (ptrdiff_t incr)

  _sbrk() (p. *26*) *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() (p. 26) 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 system_stm32g4xx.c File Reference

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

```
#include "stm32g4xx.h"
```

Macros

- #define **HSE_VALUE** 24000000U
- #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** = **HSI_VALUE**
- 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}

4.6.1 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()** (p. 7): This function is called at startup just after reset and before branch to main program. This call is made inside the "startup_stm32g4xx.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()** (p. 7): Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.

After each device reset the HSI (16 MHz) is used as system clock source. Then **SystemInit()** (p. 7) function is called, in "startup_stm32g4xx.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 | HSI

4.6.2.2 SYSCLK(Hz) | 16000000

4.6.2.3 HCLK(Hz) | 16000000

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 | 16

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 Require 48MHz for RNG | Disabled

==========

Attention

Index