My Project

Generated by Doxygen 1.12.0

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

- Stm32g4xx_system

3.1.1 Detailed Description

3.1.2 Stm32g4xx_system

Topics

- 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

3.1.2.1 Detailed Description

3.1.2.2 STM32G4xx_System_Private_Includes

Macros

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

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 24000000U
```

Value of the External oscillator in Hz

3.1.2.2.2.2 HSI_VALUE

```
#define HSI_VALUE 16000000U
```

Value of the Internal oscillator in Hz

3.1.2.3 STM32G4xx_System_Private_TypesDefinitions

3.1.2.4 STM32G4xx_System_Private_Defines

3.1.2.5 STM32G4xx_System_Private_Macros

3.1.2.6 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}

3.1.2.6.1 Detailed Description

3.1.2.7 STM32G4xx_System_Private_FunctionPrototypes

3.1.2.8 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.

3.1.2.8.1 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 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	

3.1.2.8.2.2 SystemInit()

```
void SystemInit (
            void )
```

Setup the microcontroller system.

Parameters

None	

Return values

None	

Chapter 4

File Documentation

4.1 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 MX_ADC2_Init (void)
- void HAL_ADC_MspInit (ADC_HandleTypeDef *adcHandle)
- void HAL_ADC_MspDeInit (ADC_HandleTypeDef *adcHandle)

Variables

- ADC_HandleTypeDef **hadc1**
- ADC_HandleTypeDef **hadc2**
- DMA_HandleTypeDef **hdma_adc1**

4.1.1 Detailed Description

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

Attention

4.1.2 Function Documentation

4.1.2.1 HAL_ADC_MspDeInit()

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

ADC1 GPIO Configuration PC2 ------> ADC1_IN8 PA1 ------> ADC1_IN2 PB0 ------> ADC1_IN15 PB1 ------> ADC1_IN12

ADC2 GPIO Configuration PC0 ------> ADC2_IN6 PC1 ------> ADC2_IN7 PC3 ------> ADC2_IN9 PA0 ------> ADC2_IN1

4.1.2.2 HAL_ADC_MspInit()

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

Initializes the peripherals clocks

ADC1 GPIO Configuration PC2 ------> ADC1_IN8 PA1 ------> ADC1_IN2 PB0 ------> ADC1_IN15 PB1 ------> ADC1_IN12

Initializes the peripherals clocks

ADC2 GPIO Configuration PC0 ------> ADC2_IN6 PC1 ------> ADC2_IN7 PC3 ------> ADC2_IN9 PA0 ------> ADC2_IN1

4.1.2.3 MX_ADC1_Init()

```
void MX_ADC1_Init (
            void )
```

Common config

Configure the ADC multi-mode

Configure Regular Channel

4.1.2.4 MX_ADC2_Init()

```
void MX_ADC2_Init (
            void )
```

Common config

Configure Regular Channel

4.2 dma.c File Reference

This file provides code for the configuration of all the requested memory to memory DMA transfers.

```
#include "dma.h"
```

Functions

- void MX_DMA_Init (void)

4.2.1 Detailed Description

This file provides code for the configuration of all the requested memory to memory DMA transfers.

Attention

4.2.2 Function Documentation

4.2.2.1 MX_DMA_Init()

```
void MX_DMA_Init (
             void )
```

Enable DMA controller clock

4.3 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

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 PC10 ------> USART3_TX PC11 ------> USART3_RX

4.4 main.c File Reference

: Main program body

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

Macros

- #define **UART_TX_BUFFER_SIZE** 64
- #define **UART_RX_BUFFER_SIZE** 1
- #define **CMD_BUFFER_SIZE** 64
- #define **MAX_ARGS** 9
- #define **ASCII_ENTER** 0x0D
- #define **MAX_SPEED** 1024
- #define **ADC_BUF_SIZE** 8
- #define **PULSES_PER_REVOLUTION** 1000
- #define **SAMPLE_PERIOD** 0.01

Functions

- void SystemClock_Config (void)

  System Clock Configuration.
- 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

- uint16_t ADC_buffer [ADC_BUF_SIZE]
- char **Speed_buf** [5]

4.4.1 Detailed Description

: Main program body

Attention

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

None	

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 TIM6 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

htim	: TIM handle

Return values

None	

4.4.2.3 main()

```
int main (
             void )
```

The application entry point.

Return values

int	

< Démarrage de la conversion ADC en mode DMA.

< Activation du Timer 1.

< Gestion des commandes utilisateur via le shell MCC.

4.4.2.4 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

4.4.3 Variable Documentation

4.4.3.1 ADC_buffer

```
uint16_t ADC_buffer[ADC_BUF_SIZE]
```

Buffer pour les données ADC

4.5 process.c File Reference

Implémentation des fonctions pour la gestion des commandes via un shell UART sur STM32.

```
#include "process.h"
#include "usart.h"
#include "tim.h"
#include "adc.h"
```

Macros

- #define UART_TX_BUFFER_SIZE 64
- #define UART_RX_BUFFER_SIZE 1
- #define CMD_BUFFER_SIZE 64
- #define MAX_ARGS 9
- #define ASCII_ENTER 0x0D
- #define MAX_SPEED 1024
- #define ADC_BUF_SIZE 8

Functions

- void HAL_UART_RxCpltCallback (UART_HandleTypeDef ∗huart)

  Callback pour la réception UART.
- void shell ()

  Fonction principale de gestion du shell UART.
- void speed ()

  Calcule et affiche la vitesse actuelle.
- void ADC ()

  Lit les données du capteur ADC et calcule le courant correspondant.
- void processCommand (char ∗cmd)

  Traite une commande reçue via UART et exécute l'action correspondante.

Variables

- uint16_t ADC_buffer [ADC_BUF_SIZE]
- char cmdBuffer [CMD_BUFFER_SIZE]
- int idx_cmd
- char ∗ argv [MAX_ARGS]
- int argc = 0
- int32_t ch_MCC = 0
- uint8_t former_speed = 0

4.5.1 Detailed Description

Implémentation des fonctions pour la gestion des commandes via un shell UART sur STM32.

Ce fichier contient le code permettant de gérer des commandes utilisateur pour contrôler divers périphériques via un shell interactif.

Date

27 novembre 2024

Author

MAMOUR SARR

4.5.2 Macro Definition Documentation

4.5.2.1 ADC_BUF_SIZE

```
#define ADC_BUF_SIZE 8
```

Taille du buffer ADC

4.5.2.2 ASCII_ENTER

```
#define ASCII_ENTER 0x0D
```

Code ASCII pour la touche ENTER

4.5.2.3 CMD_BUFFER_SIZE

```
#define CMD_BUFFER_SIZE 64
```

Taille maximale du buffer de commande

4.5.2.4 MAX_ARGS

```
#define MAX_ARGS 9
```

Nombre maximal d'arguments pour une commande

4.5.2.5 MAX_SPEED

```
#define MAX_SPEED 1024
```

Vitesse maximale pour le moteur

4.5.2.6 UART_RX_BUFFER_SIZE

```
#define UART_RX_BUFFER_SIZE 1
```

Taille du buffer de réception UART

4.5.2.7 UART_TX_BUFFER_SIZE

```
#define UART_TX_BUFFER_SIZE 64
```

Taille du buffer de transmission UART

4.5.3 Function Documentation

4.5.3.1 ADC()

```
void ADC ()
```

Lit les données du capteur ADC et calcule le courant correspondant.

Cette fonction convertit les données brutes de l'ADC en tension, puis en courant en utilisant les spécifications du capteur (sensibilité et offset).

4.5.3.2 HAL_UART_RxCpltCallback()

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

Callback pour la réception UART.

Cette fonction est appelée automatiquement lorsqu'une donnée est reçue via UART.

Parameters

huart	Pointeur vers la structure UART utilisée.

4.5.3.3 processCommand()

```
void processCommand (
            char * cmd)
```

Traite une commande reçue via UART et exécute l'action correspondante.

Parameters

cmd	La commande saisie par l'utilisateur sous forme de chaîne de caractères.

4.5.3.4 shell()

```
void shell ()
```

Fonction principale de gestion du shell UART.

Cette fonction traite les entrées utilisateur, exécute les commandes correspondantes et renvoie les résultats via UART.

4.5.3.5 speed()

```
void speed ()
```

Calcule et affiche la vitesse actuelle.

Cette fonction lit un encodeur pour déterminer la vitesse du moteur et affiche les résultats via UART.

4.5.4 Variable Documentation

4.5.4.1 ADC_buffer

```
uint16_t ADC_buffer[ADC_BUF_SIZE]  [extern]
```

Buffer pour les données ADC

4.5.4.2 argc

```
int argc = 0
```

Nombre d'arguments pour la commande

4.5.4.3 argv

```
char* argv[MAX_ARGS]
```

Tableau des arguments d'une commande

4.5.4.4 ch_MCC

```
int32_t ch_MCC = 0
```

État du module MCC

4.5.4.5 cmdBuffer

```
char cmdBuffer[CMD_BUFFER_SIZE]
```

Buffer pour les commandes utilisateur

4.5.4.6 former_speed

```
uint8_t former_speed = 0
```

Dernière vitesse définie

4.5.4.7 idx_cmd

```
int idx_cmd
```

Index courant dans le buffer de commande

4.6 stm32g4xx_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)

4.6.1 Detailed Description

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

Attention

4.6.2 Function Documentation

4.6.2.1 HAL_MspInit()

```
void HAL_MspInit (
            void )
```

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

4.7 stm32g4xx_hal_timebase_tim.c File Reference

HAL time base based on the hardware TIM.

```
#include "stm32g4xx_hal.h"
#include "stm32g4xx_hal_tim.h"
```

Functions

- HAL_StatusTypeDef HAL_InitTick (uint32_t TickPriority)

  This function configures the TIM6 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 **htim6**

4.7.1 Detailed Description

HAL time base based on the hardware TIM.

Attention

4.7.2 Function Documentation

4.7.2.1 HAL_InitTick()

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

This function configures the TIM6 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

TickPriority	Tick interrupt priority.

Return values

HAL	status

4.7.2.2 HAL_ResumeTick()

```
void HAL_ResumeTick (
            void )
```

Resume Tick increment.

Note

Enable the tick increment by Enabling TIM6 update interrupt.

Parameters

None	

Return values

None	

4.7.2.3 HAL_SuspendTick()

```
void HAL_SuspendTick (
             void )
```

Suspend Tick increment.

Note

Disable the tick increment by disabling TIM6 update interrupt.

Parameters

None	

Return values

None	

4.8 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 **DMA1_Channel2_IRQHandler** (void)

  This function handles DMA1 channel2 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.
- void **TIM6_DAC_IRQHandler** (void)

  This function handles TIM6 global interrupt, DAC1 and DAC3 channel underrun error interrupts.

Variables

- DMA_HandleTypeDef **hdma_adc1**
- UART_HandleTypeDef **huart2**
- TIM_HandleTypeDef **htim6**

4.8.1 Detailed Description

Interrupt Service Routines.

Attention

4.9 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

4.9.1 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

4.10 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

4.10.1 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

4.10.2 Function Documentation

4.10.2.1 _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

4.11 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}

4.11.1 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_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(): 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() function is called, in "startup_stm32g4xx.s" file, to configure the system clock before to branch to main program.

4.11.2 This file configures the system clock as follows:

4.11.2.1 System Clock source | HSI

4.11.2.2 SYSCLK(Hz) | 16000000

4.11.2.3 HCLK(Hz) | 16000000

4.11.2.4 AHB Prescaler | 1

4.11.2.5 APB1 Prescaler | 1

4.11.2.6 APB2 Prescaler | 1

4.11.2.7 PLL_M | 1

4.11.2.8 PLL_N | 16

4.11.2.9 PLL_P | 7

4.11.2.10 PLL_Q | 2

4.11.2.11 PLL_R | 2

4.11.2.12 Require 48MHz for RNG | Disabled

==========

Attention

4.12 tim.c File Reference

This file provides code for the configuration of the TIM instances.
`#include "tim.h"`

Functions

- void **MX_TIM1_Init** (void)
- void **MX_TIM3_Init** (void)
- void **HAL_TIM_Base_MspInit** (TIM_HandleTypeDef *tim_baseHandle)
- void HAL_TIM_Encoder_MspInit (TIM_HandleTypeDef *tim_encoderHandle)
- void HAL_TIM_MspPostInit (TIM_HandleTypeDef *timHandle)
- void **HAL_TIM_Base_MspDeInit** (TIM_HandleTypeDef *tim_baseHandle)
- void HAL_TIM_Encoder_MspDeInit (TIM_HandleTypeDef *tim_encoderHandle)

Variables

- TIM_HandleTypeDef **htim1**
- TIM_HandleTypeDef **htim3**

4.12.1 Detailed Description

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

Attention

4.12.2 Function Documentation

4.12.2.1 HAL_TIM_Encoder_MspDeInit()

```
void HAL_TIM_Encoder_MspDeInit (
            TIM_HandleTypeDef * tim_encoderHandle)
```

TIM3 GPIO Configuration PA4 ------> TIM3_CH2 PA6 ------> TIM3_CH1 PC8 ------> TIM3_CH3

4.12.2.2 HAL_TIM_Encoder_MspInit()

```
void HAL_TIM_Encoder_MspInit (
            TIM_HandleTypeDef * tim_encoderHandle)
```

TIM3 GPIO Configuration PA4 ------> TIM3_CH2 PA6 ------> TIM3_CH1 PC8 ------> TIM3_CH3

4.12.2.3 HAL_TIM_MspPostInit()

```
void HAL_TIM_MspPostInit (
            TIM_HandleTypeDef * timHandle)
```

TIM1 GPIO Configuration PB13 ------> TIM1_CH1N PB14 ------> TIM1_CH2N PB15 ------> TIM1_CH3N PA8 ------> TIM1_CH1 PA9 ------> TIM1_CH2 PA10 ------> TIM1_CH3

4.13 usart.c File Reference

This file provides code for the configuration of the USART instances.
`#include "usart.h"`

Functions

- void **MX_USART2_UART_Init** (void)
- void HAL_UART_MspInit (UART_HandleTypeDef ∗uartHandle)
- void HAL_UART_MspDeInit (UART_HandleTypeDef ∗uartHandle)

Variables

- UART_HandleTypeDef **huart2**

4.13.1 Detailed Description

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

Attention

4.13.2 Function Documentation

4.13.2.1 HAL_UART_MspDeInit()

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

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

4.13.2.2 HAL_UART_MspInit()

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

Initializes the peripherals clocks
USART2 GPIO Configuration PA2 ------> USART2_TX PA3 ------> USART2_RX

Index