TP Capteur - MSC

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 functions.c File Reference

: Functions used in the main program body

```
#include "functions.h"
#include "Const.h"
```

Macros

- #define **SIZE** 32

Functions

- void **Init** (I2C_HandleTypeDef ∗hi2c)

  This functions is used for I2C initialization.
- void **Measure_T** (I2C_HandleTypeDef ∗hi2c, double ∗temp)

  This function gives the temperature from the BMP module.

4.1.1 Detailed Description

: Functions used in the main program body

4.1.2 Function Documentation

4.1.2.1 Init()

```
void Init (
            I2C_HandleTypeDef * hi2c )
```

This functions is used for I2C initialization.

Parameters

in	*hi2c*	I2C from the board choosed by the user

Return values

void	

4.1.2.2 Measure_T()

```
void Measure_T (
            I2C_HandleTypeDef * hi2c,
            double * temp )
```

This function gives the temperature from the BMP module.

Parameters

in	*hi2c*	I2C from the board choosed by the user
in	*temp*	Temperature variable used to store the measure

Return values

void	but print the mesured temperature in the shell

4.2 main.c File Reference

: Main program body

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

Functions

- void **SystemClock_Config** (void)

  System Clock Configuration.
- void **I2C_Scan** ()

  This function makes a scan of the module and print the addresses from its registers.
- void **I2C_ID** ()

  This function find the ID of the module to confirm all is ok for the user.
- int **main** (void)

  The application entry point.
- void **Error_Handler** (void)

  This function is executed in case of error occurrence.

Variables

- I2C_HandleTypeDef **hi2c1**
- UART_HandleTypeDef **hlpuart1**
- uint8_t **ret**
- uint8_t **MemAddress** = 0
- uint8_t **BufferId** [25] = {0}
- uint8_t **BufferScan** [25] = {0}
- uint8_t **StartMSG** [ ] = "_______\r\n Scanne de l'I2C : \r\n\r\n"
- uint8_t **EndMSG** [ ] = "\r\n Fini ! \r\n\r\n"
- char **mess**
- double **temp** [16]

4.2.1 Detailed Description

: Main program body

Author

: CHAPUIS Clément & SUTRA Aurélien

Attention

4.2.2 Function Documentation

4.2.2.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

4.2.2.2 I2C_ID()

```
void I2C_ID ( )
```

This function find the ID of the module to confirm all is ok for the user.

Return values

void	but print the ID of the module

4.2.2.3 I2C_Scan()

```
void I2C_Scan ( )
```

This function makes a scan of the module and print the addresses from its registers.

Return values

void	but print the addresses from the module registers

4.2.2.4 main()

```
int main (
             void  )
```

The application entry point.

Return values

int	

4.2.2.5 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

4.3 stm32g4xx_hal_msp.c File Reference

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

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

Macros

- #define **PUTCHAR_PROTOTYPE** int fputc(int ch, FILE ∗f)

Functions

- void **HAL_MspInit** (void)
- void **HAL_I2C_MspInit** (I2C_HandleTypeDef ∗hi2c)

  I2C MSP Initialization This function configures the hardware resources used in this example.
- void **HAL_I2C_MspDeInit** (I2C_HandleTypeDef ∗hi2c)

  I2C 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

- UART_HandleTypeDef **hlpuart1**
- **PUTCHAR_PROTOTYPE**

  Retargets the C library printf function to the USART.
- return **ch**

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_I2C_MspDeInit()

```
void HAL_I2C_MspDeInit (
            I2C_HandleTypeDef * hi2c )
```

I2C MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

hi2c	I2C handle pointer

Return values

None	

I2C1 GPIO Configuration PA15 ------> I2C1_SCL PB7 ------> I2C1_SDA

4.3.2.2 HAL_I2C_MspInit()

```
void HAL_I2C_MspInit (
            I2C_HandleTypeDef * hi2c )
```

I2C MSP Initialization This function configures the hardware resources used in this example.

Parameters

hi2c	I2C handle pointer

Return values

None	

Initializes the peripherals clocks

I2C1 GPIO Configuration PA15 ------> I2C1_SCL PB7 ------> I2C1_SDA

4.3.2.3 HAL_MspInit()

```
void HAL_MspInit (
            void  )
```

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

4.3.2.4 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	

LPUART1 GPIO Configuration PA2 ----—> LPUART1_TX PA3 ----—> LPUART1_RX

4.3.2.5 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

LPUART1 GPIO Configuration PA2 ----—> LPUART1_TX PA3 ----—> LPUART1_RX

4.3.3 Variable Documentation

4.3.3.1 PUTCHAR_PROTOTYPE

```
PUTCHAR_PROTOTYPE
```

Initial value:

```
{


  HAL_UART_Transmit(&hlpuart1, (uint8_t *)&ch, 1, 0xFFFF)
```

Retargets the C library printf function to the USART.

Parameters

None	

Return values

None	

4.4 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.

4.4.1 Detailed Description

Interrupt Service Routines.

Attention

4.5 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

4.6 sysmem.c File Reference

STM32CubeIDE System Memory calls file.

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

Functions

- void ∗ **_sbrk** (ptrdiff_t incr)

  _sbrk() *(p. 18) 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() (p. 18) 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 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}

4.7.1 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()** (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.

4.7.2 This file configures the system clock as follows:

4.7.2.1 System Clock source | HSI

4.7.2.2 SYSCLK(Hz) | 16000000

4.7.2.3 HCLK(Hz) | 16000000

4.7.2.4 AHB Prescaler | 1

4.7.2.5 APB1 Prescaler | 1

4.7.2.6 APB2 Prescaler | 1

4.7.2.7 PLL_M | 1

4.7.2.8 PLL_N | 16

4.7.2.9 PLL_P | 7

4.7.2.10 PLL_Q | 2

4.7.2.11 PLL_R | 2

4.7.2.12 Require 48MHz for RNG | Disabled

===

Attention

Index