STM32 Project

Generated by Doxygen 1.8.13

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

File Index

2.1 File List

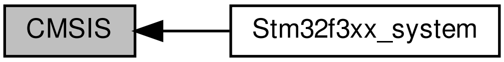
Here is a list of all files with brief descriptions:

Chapter 3

Module Documentation

3.1 CMSIS

Collaboration diagram for CMSIS:



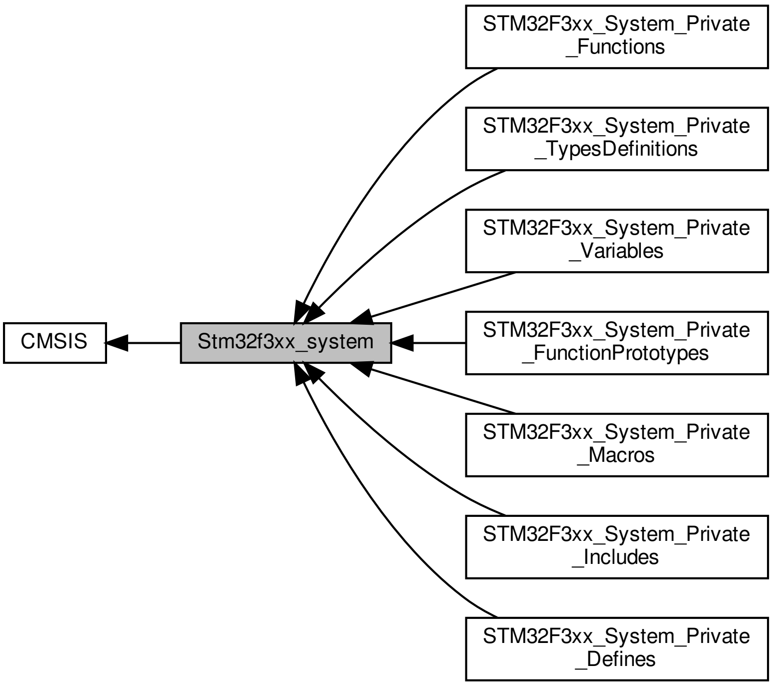
Modules

- Stm32f3xx_system

3.1.1 Detailed Description

3.2 Stm32f3xx_system

Collaboration diagram for Stm32f3xx_system:


STM32F3xx_System_Private _Functions
STM32F3xx_System_Private _TypesDefinitions
STM32F3xx_System_Private _Variables
CMSIS
Stm32f3xx_system
STM32F3xx_System_Private _FunctionPrototypes
STM32F3xx_System_Private _Macros
STM32F3xx_System_Private _Includes
STM32F3xx_System_Private _Defines

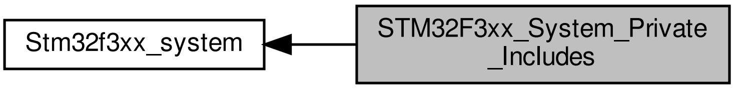
Modules

- STM32F3xx_System_Private_Includes
- STM32F3xx_System_Private_TypesDefinitions
- STM32F3xx_System_Private_Defines
- STM32F3xx_System_Private_Macros
- STM32F3xx_System_Private_Variables
- STM32F3xx_System_Private_FunctionPrototypes
- STM32F3xx_System_Private_Functions

3.2.1 Detailed Description

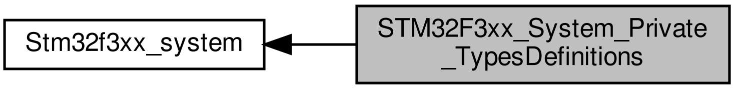
3.3 STM32F3xx_System_Private_Includes

Collaboration diagram for STM32F3xx_System_Private_Includes:



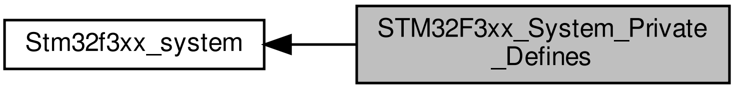
3.4 STM32F3xx_System_Private_TypesDefinitions

Collaboration diagram for STM32F3xx_System_Private_TypesDefinitions:



3.5 STM32F3xx_System_Private_Defines

Collaboration diagram for STM32F3xx_System_Private_Defines:



Macros

- #define HSE_VALUE ((uint32_t)8000000)
- #define HSI_VALUE ((uint32_t)8000000)
- #define VECT_TAB_OFFSET 0x0

3.5.1 Detailed Description

3.5.2 Macro Definition Documentation

3.5.2.1 HSE_VALUE

```
#define HSE_VALUE ((uint32_t)8000000)
```

Default value of the External oscillator in Hz. This value can be provided and adapted by the user application.

3.5.2.2 HSI_VALUE

```
#define HSI_VALUE ((uint32_t)8000000)
```

Default value of the Internal oscillator in Hz. This value can be provided and adapted by the user application.

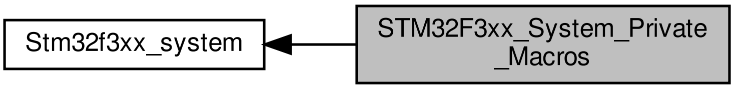
3.5.2.3 VECT_TAB_OFFSET

```
#define VECT_TAB_OFFSET 0x0
```

< Uncomment the following line if you need to relocate your vector Table in Internal SRAM. Vector Table base offset field. This value must be a multiple of 0x200.

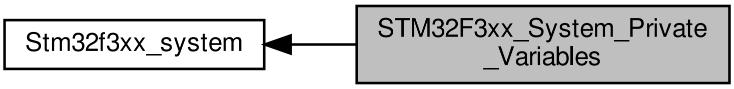
3.6 STM32F3xx_System_Private_Macros

Collaboration diagram for STM32F3xx_System_Private_Macros:



3.7 STM32F3xx_System_Private_Variables

Collaboration diagram for STM32F3xx_System_Private_Variables:



Variables

- uint32_t SystemCoreClock = 8000000
- 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}

3.7.1 Detailed Description

3.7.2 Variable Documentation

3.7.2.1 AHBPrescTable

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

3.7.2.2 APBPrescTable

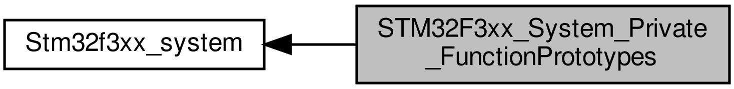
```
const uint8_t APBPrescTable[8] = {0, 0, 0, 0, 1, 2, 3, 4}
```

3.7.2.3 SystemCoreClock

```
uint32_t SystemCoreClock = 8000000
```

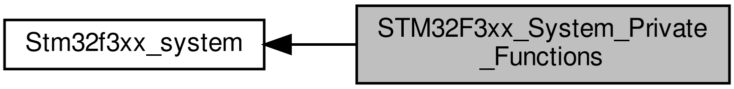
3.8 STM32F3xx_System_Private_FunctionPrototypes

Collaboration diagram for STM32F3xx_System_Private_FunctionPrototypes:



3.9 STM32F3xx_System_Private_Functions

Collaboration diagram for STM32F3xx_System_Private_Functions:



Functions

- void SystemInit (void)

  Setup the microcontroller system Initialize the FPU setting, vector table location and the PLL configuration is reset.
- 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(∗)
- 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 stm32f3xx_hal.h file (default value 8 MHz) but the real value may vary depending on the variations in voltage and temperature.

(∗∗) HSE_VALUE is a constant defined in stm32f3xx_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.

Parameters

None	

Return values

None	

3.9.2.2 SystemInit()

```
void SystemInit (
             void  )
```

Setup the microcontroller system Initialize the FPU setting, vector table location and the PLL configuration is reset.

Parameters

None	

Return values

None	

Chapter 4

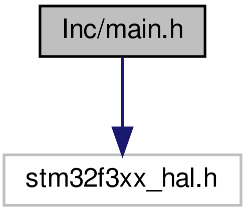
File Documentation

4.1 Inc/main.h File Reference

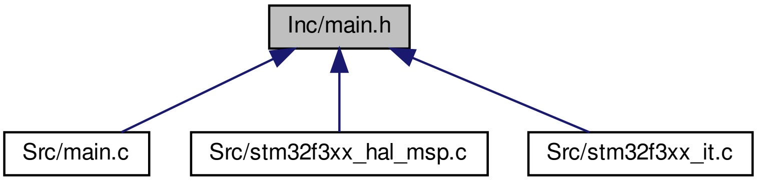
: Header for main.c file. This file contains the common defines of the application.

```
#include "stm32f3xx_hal.h"
```

Include dependency graph for main.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define B1_Pin GPIO_PIN_13
- #define B1_GPIO_Port GPIOC
- #define B1_EXTI_IRQn EXTI15_10_IRQn
- #define RCC_OSC32_OUT_Pin GPIO_PIN_15
- #define RCC_OSC32_OUT_GPIO_Port GPIOC
- #define USART_TX_Pin GPIO_PIN_2
- #define USART_TX_GPIO_Port GPIOA
- #define USART_RX_Pin GPIO_PIN_3
- #define USART_RX_GPIO_Port GPIOA
- #define LD2_Pin GPIO_PIN_5
- #define LD2_GPIO_Port GPIOA
- #define TMS_Pin GPIO_PIN_13
- #define TMS_GPIO_Port GPIOA
- #define TCK_Pin GPIO_PIN_14
- #define TCK_GPIO_Port GPIOA
- #define SWO_Pin GPIO_PIN_3
- #define SWO_GPIO_Port GPIOB

Functions

- void Error_Handler (void)

  This function is executed in case of error occurrence.

4.1.1 Detailed Description

: Header for main.c file. This file contains the common defines of the application.

Attention

4.1.2 Macro Definition Documentation

4.1.2.1 B1_EXTI_IRQn

```
#define B1_EXTI_IRQn EXTI15_10_IRQn
```

4.1.2.2 B1_GPIO_Port

```
#define B1_GPIO_Port GPIOC
```

4.1.2.3 B1_Pin

```
#define B1_Pin GPIO_PIN_13
```

4.1.2.4 LD2_GPIO_Port

```
#define LD2_GPIO_Port GPIOA
```

4.1.2.5 LD2_Pin

```
#define LD2_Pin GPIO_PIN_5
```

4.1.2.6 RCC_OSC32_OUT_GPIO_Port

```
#define RCC_OSC32_OUT_GPIO_Port GPIOC
```

4.1.2.7 RCC_OSC32_OUT_Pin

```
#define RCC_OSC32_OUT_Pin GPIO_PIN_15
```

4.1.2.8 SWO_GPIO_Port

```
#define SWO_GPIO_Port GPIOB
```

4.1.2.9 SWO_Pin

```
#define SWO_Pin GPIO_PIN_3
```

4.1.2.10 TCK_GPIO_Port

```
#define TCK_GPIO_Port GPIOA
```

4.1.2.11 TCK_Pin

```
#define TCK_Pin GPIO_PIN_14
```

4.1.2.12 TMS_GPIO_Port

```
#define TMS_GPIO_Port GPIOA
```

4.1.2.13 TMS_Pin

```
#define TMS_Pin GPIO_PIN_13
```

4.1.2.14 USART_RX_GPIO_Port

```
#define USART_RX_GPIO_Port GPIOA
```

4.1.2.15 USART_RX_Pin

```
#define USART_RX_Pin GPIO_PIN_3
```

4.1.2.16 USART_TX_GPIO_Port

```
#define USART_TX_GPIO_Port GPIOA
```

4.1.2.17 USART_TX_Pin

```
#define USART_TX_Pin GPIO_PIN_2
```

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

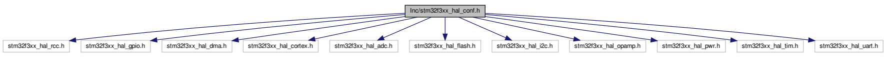
None	

4.2 Inc/stm32f3xx_hal_conf.h File Reference

HAL configuration file.

```
#include "stm32f3xx_hal_rcc.h"
#include "stm32f3xx_hal_gpio.h"
#include "stm32f3xx_hal_dma.h"
#include "stm32f3xx_hal_cortex.h"
#include "stm32f3xx_hal_adc.h"
#include "stm32f3xx_hal_flash.h"
#include "stm32f3xx_hal_i2c.h"
#include "stm32f3xx_hal_opamp.h"
#include "stm32f3xx_hal_pwr.h"
#include "stm32f3xx_hal_tim.h"
#include "stm32f3xx_hal_uart.h"
```

Include dependency graph for stm32f3xx_hal_conf.h:



Macros

- #define HAL_MODULE_ENABLED

  This is the list of modules to be used in the HAL driver.
- #define HAL_ADC_MODULE_ENABLED
- #define HAL_OPAMP_MODULE_ENABLED
- #define HAL_TIM_MODULE_ENABLED

- #define HAL_UART_MODULE_ENABLED
- #define HAL_GPIO_MODULE_ENABLED
- #define HAL_DMA_MODULE_ENABLED
- #define HAL_RCC_MODULE_ENABLED
- #define HAL_FLASH_MODULE_ENABLED
- #define HAL_PWR_MODULE_ENABLED
- #define HAL_CORTEX_MODULE_ENABLED
- #define HAL_I2C_MODULE_ENABLED
- #define HSE_VALUE ((uint32_t)8000000)

  Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).
- #define HSE_STARTUP_TIMEOUT ((uint32_t)100)

  In the following line adjust the External High Speed oscillator (HSE) Startup Timeout value.
- #define HSI_VALUE ((uint32_t)8000000)

  Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).
- #define HSI_STARTUP_TIMEOUT ((uint32_t)5000)

  In the following line adjust the Internal High Speed oscillator (HSI) Startup Timeout value.
- #define LSI_VALUE ((uint32_t)40000)

  Internal Low Speed oscillator (LSI) value.
- #define LSE_VALUE ((uint32_t)32768)

  External Low Speed oscillator (LSE) value.
- #define LSE_STARTUP_TIMEOUT ((uint32_t)5000)

  Time out for LSE start up value in ms.
- #define EXTERNAL_CLOCK_VALUE ((uint32_t)8000000)

  External clock source for I2S peripheral This value is used by the I2S HAL module to compute the I2S clock source frequency, this source is inserted directly through I2S_CKIN pad.
- #define VDD_VALUE ((uint32_t)3300)

  This is the HAL system configuration section.
- #define TICK_INT_PRIORITY ((uint32_t)0)
- #define USE_RTOS 0
- #define PREFETCH_ENABLE 1
- #define INSTRUCTION_CACHE_ENABLE 0
- #define DATA_CACHE_ENABLE 0
- #define USE_SPI_CRC 0U

  Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.
- #define assert_param(expr) ((void)0U)

  Include module's header file.

4.2.1 Detailed Description

HAL configuration file.

Attention

4.2.2 Macro Definition Documentation

4.2.2.1 assert_param

```
#define assert_param(
             expr ) ((void)0U)
```

Include module's header file.

4.2.2.2 DATA_CACHE_ENABLE

```
#define DATA_CACHE_ENABLE 0
```

4.2.2.3 EXTERNAL_CLOCK_VALUE

```
#define EXTERNAL_CLOCK_VALUE ((uint32_t)8000000)
```

External clock source for I2S peripheral This value is used by the I2S HAL module to compute the I2S clock source frequency, this source is inserted directly through I2S_CKIN pad.

- External clock generated through external PLL component on EVAL 303 (based on MCO or crystal)
- External clock not generated on EVAL 373Value of the External oscillator in Hz

4.2.2.4 HAL_ADC_MODULE_ENABLED

```
#define HAL_ADC_MODULE_ENABLED
```

4.2.2.5 HAL_CORTEX_MODULE_ENABLED

```
#define HAL_CORTEX_MODULE_ENABLED
```

4.2.2.6 HAL_DMA_MODULE_ENABLED

```
#define HAL_DMA_MODULE_ENABLED
```

4.2.2.7 HAL_FLASH_MODULE_ENABLED

```
#define HAL_FLASH_MODULE_ENABLED
```

4.2.2.8 HAL_GPIO_MODULE_ENABLED

```
#define HAL_GPIO_MODULE_ENABLED
```

4.2.2.9 HAL_I2C_MODULE_ENABLED

```
#define HAL_I2C_MODULE_ENABLED
```

4.2.2.10 HAL_MODULE_ENABLED

```
#define HAL_MODULE_ENABLED
```

This is the list of modules to be used in the HAL driver.

4.2.2.11 HAL_OPAMP_MODULE_ENABLED

```
#define HAL_OPAMP_MODULE_ENABLED
```

4.2.2.12 HAL_PWR_MODULE_ENABLED

```
#define HAL_PWR_MODULE_ENABLED
```

4.2.2.13 HAL_RCC_MODULE_ENABLED

```
#define HAL_RCC_MODULE_ENABLED
```

4.2.2.14 HAL_TIM_MODULE_ENABLED

```
#define HAL_TIM_MODULE_ENABLED
```

4.2.2.15 HAL_UART_MODULE_ENABLED

```
#define HAL_UART_MODULE_ENABLED
```

4.2.2.16 HSE_STARTUP_TIMEOUT

```
#define HSE_STARTUP_TIMEOUT ((uint32_t)100)
```

In the following line adjust the External High Speed oscillator (HSE) Startup Timeout value.

Time out for HSE start up, in ms

4.2.2.17 HSE_VALUE

```
#define HSE_VALUE ((uint32_t)8000000)
```

Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).

Value of the External oscillator in Hz

4.2.2.18 HSI_STARTUP_TIMEOUT

```
#define HSI_STARTUP_TIMEOUT ((uint32_t)5000)
```

In the following line adjust the Internal High Speed oscillator (HSI) Startup Timeout value.

Time out for HSI start up

4.2.2.19 HSI_VALUE

```
#define HSI_VALUE ((uint32_t)8000000)
```

Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).

Value of the Internal oscillator in Hz

4.2.2.20 INSTRUCTION_CACHE_ENABLE

```
#define INSTRUCTION_CACHE_ENABLE 0
```

4.2.2.21 LSE_STARTUP_TIMEOUT

```
#define LSE_STARTUP_TIMEOUT ((uint32_t)5000)
```

Time out for LSE start up value in ms.

Time out for LSE start up, in ms

4.2.2.22 LSE_VALUE

```
#define LSE_VALUE ((uint32_t)32768)
```

External Low Speed oscillator (LSE) value.

< Value of the Internal Low Speed oscillator in Hz The real value may vary depending on the variations in voltage and temperature. Value of the External Low Speed oscillator in Hz

4.2.2.23 LSI_VALUE

```
#define LSI_VALUE ((uint32_t)40000)
```

Internal Low Speed oscillator (LSI) value.

4.2.2.24 PREFETCH_ENABLE

```
#define PREFETCH_ENABLE 1
```

4.2.2.25 TICK_INT_PRIORITY

```
#define TICK_INT_PRIORITY ((uint32_t)0)
```

tick interrupt priority (lowest by default)

4.2.2.26 USE_RTOS

```
#define USE_RTOS 0
```

4.2.2.27 USE_SPI_CRC

```
#define USE_SPI_CRC 0U
```

Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.

4.2.2.28 VDD_VALUE

```
#define VDD_VALUE ((uint32_t)3300)
```

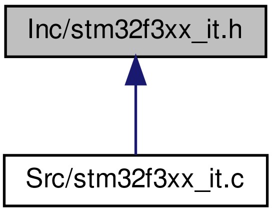
This is the HAL system configuration section.

Value of VDD in mv

4.3 Inc/stm32f3xx_it.h File Reference

This file contains the headers of the interrupt handlers.

This graph shows which files directly or indirectly include this file:



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 DMA1_Channel1_IRQHandler (void)

  This function handles DMA1 channel1 global interrupt.
- void ADC1_2_IRQHandler (void)

  This function handles ADC1 and ADC2 interrupts.
- void TIM1_UP_TIM16_IRQHandler (void)

  This function handles TIM1 update and TIM16 interrupts.
- void I2C1_EV_IRQHandler (void)

  This function handles I2C1 event global interrupt / I2C1 wake-up interrupt through EXT line 23.
- void I2C1_ER_IRQHandler (void)

  This function handles I2C1 error interrupt.
- void USART2_IRQHandler (void)

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

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

4.3.1 Detailed Description

This file contains the headers of the interrupt handlers.

Attention

4.3.2 Function Documentation

4.3.2.1 ADC1_2_IRQHandler()

```
void ADC1_2_IRQHandler (
             void  )
```

This function handles ADC1 and ADC2 interrupts.

4.3.2.2 BusFault_Handler()

```
void BusFault_Handler (
             void  )
```

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

4.3.2.3 DebugMon_Handler()

```
void DebugMon_Handler (
             void  )
```

This function handles Debug monitor.

4.3.2.4 DMA1_Channel1_IRQHandler()

```
void DMA1_Channel1_IRQHandler (
            void  )
```

This function handles DMA1 channel1 global interrupt.

4.3.2.5 EXTI15_10_IRQHandler()

```
void EXTI15_10_IRQHandler (
            void  )
```

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

4.3.2.6 HardFault_Handler()

```
void HardFault_Handler (
            void  )
```

This function handles Hard fault interrupt.

4.3.2.7 I2C1_ER_IRQHandler()

```
void I2C1_ER_IRQHandler (
            void  )
```

This function handles I2C1 error interrupt.

4.3.2.8 I2C1_EV_IRQHandler()

```
void I2C1_EV_IRQHandler (
            void  )
```

This function handles I2C1 event global interrupt / I2C1 wake-up interrupt through EXT line 23.

4.3.2.9 MemManage_Handler()

```
void MemManage_Handler (
            void  )
```

This function handles Memory management fault.

4.3.2.10 NMI_Handler()

```
void NMI_Handler (
            void  )
```

This function handles Non maskable interrupt.

4.3.2.11 PendSV_Handler()

```
void PendSV_Handler (
            void  )
```

This function handles Pendable request for system service.

4.3.2.12 SVC_Handler()

```
void SVC_Handler (
            void  )
```

This function handles System service call via SWI instruction.

4.3.2.13 SysTick_Handler()

```
void SysTick_Handler (
            void  )
```

This function handles System tick timer.

4.3.2.14 TIM1_UP_TIM16_IRQHandler()

```
void TIM1_UP_TIM16_IRQHandler (
            void  )
```

This function handles TIM1 update and TIM16 interrupts.

4.3.2.15 UsageFault_Handler()

```
void UsageFault_Handler (
            void  )
```

This function handles Undefined instruction or illegal state.

4.3.2.16 USART2_IRQHandler()

```
void USART2_IRQHandler (
            void  )
```

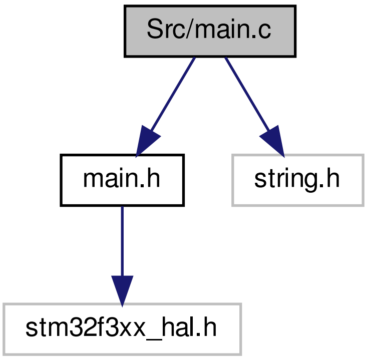
This function handles USART2 global interrupt / USART2 wake-up interrupt through EXT line 26.

4.4 Src/main.c File Reference

: Main program body

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

Include dependency graph for main.c:



Macros

- #define BUFFER_SIZE 32
- #define DATA_SIZE 2
- #define IN_SIZE 32
- #define INA219_ADDRESS (0x40<<1)
- #define INA219_REG_CONFIG (0x00)
- #define INA219_REG_SHUNTVOLTAGE (0x01)
- #define INA219_REG_BUSVOLTAGE (0x02)
- #define INA219_REG_POWER (0x03)
- #define INA219_REG_CURRENT (0x04)
- #define INA219_REG_CALIBRATION (0x05)

Functions

- void SystemClock_Config (void)

  System Clock Configuration.
- void setCalibration_16V_400mA (void)

  Sets up INA219 configuration (0x019F) and calibration (0x2000) registers.
- float getCurrent_mA (uint16_t value)

  Calculates current value.
- float getPower_mW (uint16_t value)

  Calculates power value.
- float getBusVoltage_V (uint16_t value)

  Calculates bus voltage.
- float getShuntVoltage_mV (uint16_t value)

  Calculates shunt voltage.
- int getCurrentDivider_mA (void)

  Getter for current divider.
- int getPowerMultiplier_mW (void)

  Getter for power multiplier.
- int main (void)

  The application entry point.
- void Error_Handler (void)

  This function is executed in case of error occurrence.

Variables

- ADC_HandleTypeDef hadc1
- ADC_HandleTypeDef hadc2
- DMA_HandleTypeDef hdma_adc1
- I2C_HandleTypeDef hi2c1
- OPAMP_HandleTypeDef hopamp2
- TIM_HandleTypeDef htim1
- UART_HandleTypeDef huart2
- uint8_t buffer [BUFFER_SIZE]
- uint8_t data [DATA_SIZE]
- uint8_t uart1 [IN_SIZE]
- uint8_t uart2 [IN_SIZE]
- uint8_t ∗ uart_ptr
- uint16_t adc_val
- uint32_t ina219_currentDivider_mA
- uint32_t ina219_powerMultiplier_mW

4.4.1 Detailed Description

: Main program body

Attention

4.4.2 Macro Definition Documentation

4.4.2.1 BUFFER_SIZE

```
#define BUFFER_SIZE 32
```

4.4.2.2 DATA_SIZE

```
#define DATA_SIZE 2
```

4.4.2.3 IN_SIZE

```
#define IN_SIZE 32
```

4.4.2.4 INA219_ADDRESS

```
#define INA219_ADDRESS (0x40<<1)
```

4.4.2.5 INA219_REG_BUSVOLTAGE

```
#define INA219_REG_BUSVOLTAGE (0x02)
```

4.4.2.6 INA219_REG_CALIBRATION

```
#define INA219_REG_CALIBRATION (0x05)
```

4.4.2.7 INA219_REG_CONFIG

```
#define INA219_REG_CONFIG (0x00)
```

4.4.2.8 INA219_REG_CURRENT

```
#define INA219_REG_CURRENT (0x04)
```

4.4.2.9 INA219_REG_POWER

```
#define INA219_REG_POWER (0x03)
```

4.4.2.10 INA219_REG_SHUNTVOLTAGE

```
#define INA219_REG_SHUNTVOLTAGE (0x01)
```

4.4.3 Function Documentation

4.4.3.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

4.4.3.2 getBusVoltage_V()

```
float getBusVoltage_V (
            uint16_t value )
```

Calculates bus voltage.

Parameters

2	bytes raw value

Returns

returns measured bus voltage in V

4.4.3.3 getCurrent_mA()

```
float getCurrent_mA (
            uint16_t value )
```

Calculates current value.

Parameters

2	bytes raw value

Returns

returns measured current in mA

4.4.3.4 getCurrentDivider_mA()

```
int getCurrentDivider_mA (
            void  )
```

Getter for current divider.

Returns

returns current divider in mA

4.4.3.5 getPower_mW()

```
float getPower_mW (
            uint16_t value )
```

Calculates power value.

Parameters

2	bytes raw value

Returns

returns measured power in mW

4.4.3.6 getPowerMultiplier_mW()

```
int getPowerMultiplier_mW (
             void  )
```

Getter for power multiplier.

Returns

returns power multiplier in mW

4.4.3.7 getShuntVoltage_mV()

```
float getShuntVoltage_mV (
             uint16_t value )
```

Calculates shunt voltage.

Parameters

2	bytes raw value

Returns

returns measured ahunt voltage in mV

4.4.3.8 main()

```
int main (
             void  )
```

The application entry point.

Return values

int	

4.4.3.9 setCalibration_16V_400mA()

```
void setCalibration_16V_400mA (
             void  )
```

Sets up INA219 configuration (0x019F) and calibration (0x2000) registers.

4.4.3.10 SystemClock_Config()

```
void SystemClock_Config (
             void  )
```

System Clock Configuration.

Return values

None	

Initializes the CPU, AHB and APB busses clocks

Initializes the CPU, AHB and APB busses clocks

4.4.4 Variable Documentation

4.4.4.1 adc_val

```
uint16_t adc_val
```

4.4.4.2 buffer

```
uint8_t buffer[BUFFER_SIZE]
```

4.4.4.3 data

```
uint8_t data[DATA_SIZE]
```

4.4.4.4 hadc1

```
ADC_HandleTypeDef hadc1
```

4.4.4.5 hadc2

```
ADC_HandleTypeDef hadc2
```

4.4.4.6 hdma_adc1

```
DMA_HandleTypeDef hdma_adc1
```

File Name : stm32f3xx_hal_msp.c Description : This file provides code for the MSP Initialization and de-Initialization codes.

Attention

4.4.4.7 hi2c1

```
I2C_HandleTypeDef hi2c1
```

4.4.4.8 hopamp2

```
OPAMP_HandleTypeDef hopamp2
```

4.4.4.9 htim1

```
TIM_HandleTypeDef htim1
```

4.4.4.10 huart2

```
UART_HandleTypeDef huart2
```

4.4.4.11 ina219_currentDivider_mA

```
uint32_t ina219_currentDivider_mA
```

4.4.4.12 ina219_powerMultiplier_mW

```
uint32_t ina219_powerMultiplier_mW
```

4.4.4.13 uart1

```
uint8_t uart1[IN_SIZE]
```

4.4.4.14 uart2

```
uint8_t uart2[IN_SIZE]
```

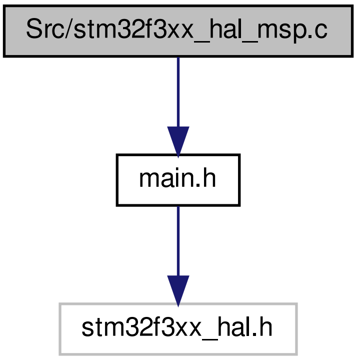
4.4.4.15 uart_ptr

```
uint8_t* uart_ptr
```

4.5 Src/stm32f3xx_hal_msp.c File Reference

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

Include dependency graph for stm32f3xx_hal_msp.c:



Functions

- 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_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_OPAMP_MspInit (OPAMP_HandleTypeDef ∗hopamp)

  OPAMP MSP Initialization This function configures the hardware resources used in this example.
- void HAL_OPAMP_MspDeInit (OPAMP_HandleTypeDef ∗hopamp)

  OPAMP 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_Base_MspDeInit (TIM_HandleTypeDef ∗htim_base)

  TIM_Base 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

4.5.1 Function Documentation

4.5.1.1 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

Uncomment the line below to disable the "ADC1_2_IRQn" interrupt Be aware, disabling shared interrupt may affect other IPs

ADC2 GPIO Configuration PC2 ——> ADC2_IN8

Uncomment the line below to disable the "ADC1_2_IRQn" interrupt Be aware, disabling shared interrupt may affect other IPs

4.5.1.2 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	

ADC1 GPIO Configuration PA0 ——> ADC1_IN1

ADC2 GPIO Configuration PC2 ---—> ADC2_IN8

4.5.1.3 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

4.5.1.4 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	

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

4.5.1.5 HAL_MspInit()

```
void HAL_MspInit (
            void )
```

Initializes the Global MSP.

4.5.1.6 HAL_OPAMP_MspDeInit()

```
void HAL_OPAMP_MspDeInit (
            OPAMP_HandleTypeDef * hopamp )
```

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

Parameters

hopamp	OPAMP handle pointer

Return values

None	

OPAMP2 GPIO Configuration PA6 ------> OPAMP2_VOUT PA7 ------> OPAMP2_VINP PC5 ------> OPAMP2_↩ VINM

4.5.1.7 HAL_OPAMP_MspInit()

```
void HAL_OPAMP_MspInit (
            OPAMP_HandleTypeDef * hopamp )
```

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

Parameters

hopamp	OPAMP handle pointer

Return values

None	

OPAMP2 GPIO Configuration PA6 ------> OPAMP2_VOUT PA7 ------> OPAMP2_VINP PC5 ------> OPAMP2_↩ VINM

4.5.1.8 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	

4.5.1.9 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	

4.5.1.10 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

4.5.1.11 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	

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

4.5.2 Variable Documentation

4.5.2.1 hdma_adc1

```
DMA_HandleTypeDef hdma_adc1
```

File Name : stm32f3xx_hal_msp.c Description : This file provides code for the MSP Initialization and de-Initialization codes.

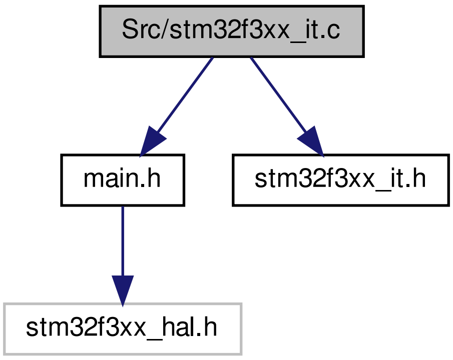
Attention

4.6 Src/stm32f3xx_it.c File Reference

Interrupt Service Routines.

```
#include "main.h"
#include "stm32f3xx_it.h"
```

Include dependency graph for stm32f3xx_it.c:



Macros

- #define BUFFER_SIZE 32
- #define DATA_SIZE 2
- #define IN_SIZE 32
- #define DMA_SIZE 4
- #define INA219_ADDRESS (0x40<<1)
- #define INA219_REG_CONFIG (0x00)
- #define INA219_REG_SHUNTVOLTAGE (0x01)
- #define INA219_REG_BUSVOLTAGE (0x02)
- #define INA219_REG_POWER (0x03)
- #define INA219_REG_CURRENT (0x04)
- #define INA219_REG_CALIBRATION (0x05)

Enumerations

- enum COMMAND {
  CMD_SHUNT_VOLTAGE = 0x01, CMD_BUS_VOLTAGE = 0x02, CMD_POWER = 0x03, CMD_CURRENT = 0x04,
  CMD_STOP = 0x05, CMD_ADC = 0x06, CMD_UNKNOWN = 0x07 }
- enum MEASUREMENT { MEASURE_INA219 = 0x01, MEASURE_CUSTOM = 0x02 }

Functions

- float getCurrent_mA (uint16_t value)

  Calculates current value.
- float getPower_mW (uint16_t value)

  Calculates power value.
- float getBusVoltage_V (uint16_t value)

  Calculates bus voltage.
- float getShuntVoltage_mV (uint16_t value)

  Calculates shunt voltage.
- void cmdParser (uint8_t ∗in, uint8_t cnt)

  Handles received commands.
- void setupTimerDMA ()

  Sets up timer counter and frequency for DMA.
- void setupTimerIRQ ()

  Sets up timer counter and frequency for IRQ.
- void convertToCommand (uint8_t ∗in)

  Tries to convert received message to command and obtains data.
- void cmdI2C ()

  Triggers INA219 actions depending on cmd variable.
- void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)

  Switches way of measurements.
- void HAL_UART_RxCpltCallback (UART_HandleTypeDef ∗huart)

  Handles receiving bytes on UART.
- void HAL_ADC_ConvCpltCallback (ADC_HandleTypeDef ∗hadc)

  Gets measurement values and send them back to PC via UART.
- void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef ∗htim)

  Triggers measurements.
- void HAL_I2C_MasterRxCpltCallback (I2C_HandleTypeDef ∗hi2c)

Handles received bytes on I2C.

- void HAL_I2C_MasterTxCpltCallback (I2C_HandleTypeDef ∗hi2c)
- void HAL_I2C_ErrorCallback (I2C_HandleTypeDef ∗hi2c)

  Sends errors.
- void HAL_UART_ErrorCallback (UART_HandleTypeDef ∗huart)

  Sends errors.
- 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 DMA1_Channel1_IRQHandler (void)

  This function handles DMA1 channel1 global interrupt.
- void ADC1_2_IRQHandler (void)

  This function handles ADC1 and ADC2 interrupts.
- void TIM1_UP_TIM16_IRQHandler (void)

  This function handles TIM1 update and TIM16 interrupts.
- void I2C1_EV_IRQHandler (void)

  This function handles I2C1 event global interrupt / I2C1 wake-up interrupt through EXT line 23.
- void I2C1_ER_IRQHandler (void)

  This function handles I2C1 error interrupt.
- void USART2_IRQHandler (void)

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

  This function handles EXTI line[15:10] interrupts.
- void HAL_UART_TxCpltCallback (UART_HandleTypeDef ∗huart)

  Reset tx_flag when transmit completed.

Variables

- uint8_t data [DATA_SIZE]
- uint8_t buffer [BUFFER_SIZE]
- uint8_t uart1 [IN_SIZE]
- uint8_t uart2 [IN_SIZE]
- uint8_t ∗ uart_ptr
- uint16_t adc_val
- volatile uint16_t cnt_tim
- volatile uint8_t tx_flag

- volatile int _cnt
- volatile int _freq
- uint16_t adc_buffer [DMA_SIZE]
- enum COMMAND cmd
- enum MEASUREMENT measurement = MEASURE_INA219
- DMA_HandleTypeDef hdma_adc1
- ADC_HandleTypeDef hadc1
- ADC_HandleTypeDef hadc2
- I2C_HandleTypeDef hi2c1
- TIM_HandleTypeDef htim1
- UART_HandleTypeDef huart2

4.6.1 Detailed Description

Interrupt Service Routines.

Attention

4.6.2 Macro Definition Documentation

4.6.2.1 BUFFER_SIZE

```
#define BUFFER_SIZE 32
```

4.6.2.2 DATA_SIZE

```
#define DATA_SIZE 2
```

4.6.2.3 DMA_SIZE

```
#define DMA_SIZE 4
```

4.6.2.4 IN_SIZE

```
#define IN_SIZE 32
```

4.6.2.5 INA219_ADDRESS

```
#define INA219_ADDRESS (0x40<<1)
```

4.6.2.6 INA219_REG_BUSVOLTAGE

```
#define INA219_REG_BUSVOLTAGE (0x02)
```

4.6.2.7 INA219_REG_CALIBRATION

```
#define INA219_REG_CALIBRATION (0x05)
```

4.6.2.8 INA219_REG_CONFIG

```
#define INA219_REG_CONFIG (0x00)
```

4.6.2.9 INA219_REG_CURRENT

```
#define INA219_REG_CURRENT (0x04)
```

4.6.2.10 INA219_REG_POWER

```
#define INA219_REG_POWER (0x03)
```

4.6.2.11 INA219_REG_SHUNTVOLTAGE

```
#define INA219_REG_SHUNTVOLTAGE (0x01)
```

4.6.3 Enumeration Type Documentation

4.6.3.1 COMMAND

```
enum COMMAND
```

Enumerator

CMD_SHUNT_VOLTAGE	
CMD_BUS_VOLTAGE	
CMD_POWER	
CMD_CURRENT	
CMD_STOP	
CMD_ADC	
CMD_UNKNOWN	

4.6.3.2 MEASUREMENT

```
enum MEASUREMENT
```

Enumerator

MEASURE_INA219	
MEASURE_CUSTOM	

4.6.4 Function Documentation

4.6.4.1 ADC1_2_IRQHandler()

```
void ADC1_2_IRQHandler (
            void  )
```

This function handles ADC1 and ADC2 interrupts.

4.6.4.2 BusFault_Handler()

```
void BusFault_Handler (
             void  )
```

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

4.6.4.3 cmdI2C()

```
void cmdI2C ( )
```

Triggers INA219 actions depending on cmd variable.

4.6.4.4 cmdParser()

```
void cmdParser (
             uint8_t * in,
             uint8_t cnt )
```

Handles received commands.

Parameters

uint8← _t	buffer

4.6.4.5 convertToCommand()

```
void convertToCommand (
             uint8_t * in )
```

Tries to convert received message to command and obtains data.

Parameters

uint8← _t	buffer

4.6.4.6 DebugMon_Handler()

```
void DebugMon_Handler (
            void )
```

This function handles Debug monitor.

4.6.4.7 DMA1_Channel1_IRQHandler()

```
void DMA1_Channel1_IRQHandler (
            void )
```

This function handles DMA1 channel1 global interrupt.

4.6.4.8 EXTI15_10_IRQHandler()

```
void EXTI15_10_IRQHandler (
            void )
```

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

4.6.4.9 getBusVoltage_V()

```
float getBusVoltage_V (
            uint16_t value )
```

Calculates bus voltage.

Parameters

2	bytes raw value

Returns

returns measured bus voltage in V

4.6.4.10 getCurrent_mA()

```
float getCurrent_mA (
            uint16_t value )
```

Calculates current value.

Parameters

2	bytes raw value

Returns

returns measured current in mA

4.6.4.11 getPower_mW()

```
float getPower_mW (
            uint16_t value )
```

Calculates power value.

Parameters

2	bytes raw value

Returns

returns measured power in mW

4.6.4.12 getShuntVoltage_mV()

```
float getShuntVoltage_mV (
            uint16_t value )
```

Calculates shunt voltage.

Parameters

2	bytes raw value

Returns

returns measured ahunt voltage in mV

4.6.4.13 HAL_ADC_ConvCpltCallback()

```
void HAL_ADC_ConvCpltCallback (
            ADC_HandleTypeDef * hadc )
```

Gets measurement values and send them back to PC via UART.

Parameters

ADC_HandleTypeDef	

4.6.4.14 HAL_GPIO_EXTI_Callback()

```
void HAL_GPIO_EXTI_Callback (
             uint16_t GPIO_Pin )
```

Switches way of measurements.

Parameters

GPIO_Pin	

4.6.4.15 HAL_I2C_ErrorCallback()

```
void HAL_I2C_ErrorCallback (
             I2C_HandleTypeDef * hi2c )
```

Sends errors.

Parameters

I2C_HandleTypeDef	

4.6.4.16 HAL_I2C_MasterRxCpltCallback()

```
void HAL_I2C_MasterRxCpltCallback (
             I2C_HandleTypeDef * hi2c )
```

Handles received bytes on I2C.

Parameters

I2C_HandleTypeDef	

4.6.4.17 HAL_I2C_MasterTxCpltCallback()

```
void HAL_I2C_MasterTxCpltCallback (
             I2C_HandleTypeDef * hi2c )
```

4.6.4.18 HAL_TIM_PeriodElapsedCallback()

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

Triggers measurements.

Parameters

TIM_HandleTypeDef	

4.6.4.19 HAL_UART_ErrorCallback()

```
void HAL_UART_ErrorCallback (
             UART_HandleTypeDef * huart )
```

Sends errors.

Parameters

UART_HandleTypeDef	

4.6.4.20 HAL_UART_RxCpltCallback()

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

Handles receiving bytes on UART.

Parameters

UART_HandleTypeDef	

4.6.4.21 HAL_UART_TxCpltCallback()

```
void HAL_UART_TxCpltCallback (
            UART_HandleTypeDef * huart )
```

Reset tx_flag when transmit completed.

Parameters

UART_HandleTypeDef	

4.6.4.22 HardFault_Handler()

```
void HardFault_Handler (
            void )
```

This function handles Hard fault interrupt.

4.6.4.23 I2C1_ER_IRQHandler()

```
void I2C1_ER_IRQHandler (
            void )
```

This function handles I2C1 error interrupt.

4.6.4.24 I2C1_EV_IRQHandler()

```
void I2C1_EV_IRQHandler (
            void )
```

This function handles I2C1 event global interrupt / I2C1 wake-up interrupt through EXT line 23.

4.6.4.25 MemManage_Handler()

```
void MemManage_Handler (
            void )
```

This function handles Memory management fault.

4.6.4.26 NMI_Handler()

```
void NMI_Handler (
             void  )
```

This function handles Non maskable interrupt.

4.6.4.27 PendSV_Handler()

```
void PendSV_Handler (
             void  )
```

This function handles Pendable request for system service.

4.6.4.28 setupTimerDMA()

```
void setupTimerDMA ( )
```

Sets up timer counter and frequency for DMA.

4.6.4.29 setupTimerIRQ()

```
void setupTimerIRQ ( )
```

Sets up timer counter and frequency for IRQ.

4.6.4.30 SVC_Handler()

```
void SVC_Handler (
             void  )
```

This function handles System service call via SWI instruction.

4.6.4.31 SysTick_Handler()

```
void SysTick_Handler (
             void  )
```

This function handles System tick timer.

4.6.4.32 TIM1_UP_TIM16_IRQHandler()

```
void TIM1_UP_TIM16_IRQHandler (
            void  )
```

This function handles TIM1 update and TIM16 interrupts.

4.6.4.33 UsageFault_Handler()

```
void UsageFault_Handler (
            void  )
```

This function handles Undefined instruction or illegal state.

4.6.4.34 USART2_IRQHandler()

```
void USART2_IRQHandler (
            void  )
```

This function handles USART2 global interrupt / USART2 wake-up interrupt through EXT line 26.

4.6.5 Variable Documentation

4.6.5.1 _cnt

```
volatile int _cnt
```

4.6.5.2 _freq

```
volatile int _freq
```

4.6.5.3 adc_buffer

```
uint16_t adc_buffer[DMA_SIZE]
```

4.6.5.4 adc_val

```
uint16_t adc_val
```

4.6.5.5 buffer

```
uint8_t buffer[BUFFER_SIZE]
```

4.6.5.6 cmd

```
enum COMMAND cmd
```

4.6.5.7 cnt_tim

```
volatile uint16_t cnt_tim
```

4.6.5.8 data

```
uint8_t data[DATA_SIZE]
```

4.6.5.9 hadc1

```
ADC_HandleTypeDef hadc1
```

4.6.5.10 hadc2

```
ADC_HandleTypeDef hadc2
```

4.6.5.11 hdma_adc1

```
DMA_HandleTypeDef hdma_adc1
```

File Name : stm32f3xx_hal_msp.c Description : This file provides code for the MSP Initialization and de-Initialization codes.

Attention

4.6.5.12 hi2c1

```
I2C_HandleTypeDef hi2c1
```

4.6.5.13 htim1

```
TIM_HandleTypeDef htim1
```

4.6.5.14 huart2

```
UART_HandleTypeDef huart2
```

4.6.5.15 measurement

```
enum MEASUREMENT measurement = MEASURE_INA219
```

4.6.5.16 tx_flag

```
volatile uint8_t tx_flag
```

4.6.5.17 uart1

```
uint8_t uart1[IN_SIZE]
```

4.6.5.18 uart2

```
uint8_t uart2[IN_SIZE]
```

4.6.5.19 uart_ptr

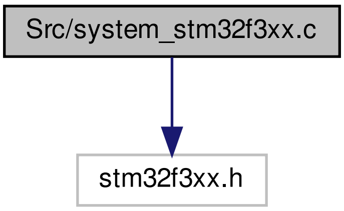
```
uint8_t* uart_ptr
```

4.7 Src/system_stm32f3xx.c File Reference

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

```
#include "stm32f3xx.h"
```

Include dependency graph for system_stm32f3xx.c:



Macros

- #define HSE_VALUE ((uint32_t)8000000)
- #define HSI_VALUE ((uint32_t)8000000)
- #define VECT_TAB_OFFSET 0x0

Functions

- void SystemInit (void)

  Setup the microcontroller system Initialize the FPU setting, vector table location and the PLL configuration is reset.
- 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 = 8000000
- 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

1. 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_stm32f3xx.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.
2. After each device reset the HSI (8 MHz) is used as system clock source. Then SystemInit() function is called, in "startup_stm32f3xx.s" file, to configure the system clock before to branch to main program.

3. This file configures the system clock as follows:

Supported STM32F3xx device

System Clock source | HSI

SYSCLK(Hz) | 8000000

HCLK(Hz) | 8000000

AHB Prescaler | 1

APB2 Prescaler | 1

APB1 Prescaler | 1

USB Clock | DISABLE

==

Attention