Systick Driver

Version 1.0.0

Generated by Doxygen 1.8.13

Contents

Chapter 1

Systick Driver

A general wrapper around systick and systick-like features to allow for general timekeeping and timeout functionality to other elements of the HAL.

Note:

The systick and sysclock elements of the HAL are some of the most difficult to use and test, and are generally wrappers around MCU vendor created functions, or reimplementations thereof with more freedom (but **STRONG** recommendations). If you have ANY doubts about my implementations or experience issues, redirect the interface targets to HALs created by the vendors.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 systick_config_t Struct Reference

```
#include <systick_stm32f411_config.h>
```

Data Fields

- systick_enabled_t enable_systick
- uint32_t tick_freq_khz
- systick_interrupt_t enable_systick_interrupt
- systick_clock_source_t clock_source

4.1.1 Detailed Description

Struct containing relevant configuration data to enable the systick

4.1.2 Field Documentation

4.1.2.1 clock_source

```
systick_clock_source_t clock_source
```

The systick clock source. Recommended value is SYSTICK_INTERNAL_CLOCK

4.1.2.2 enable_systick

```
systick_enabled_t enable_systick
```

Whether or not the systick should be enabled. Recommended value is SYSTICK_ENABLED

4.1.2.3 enable_systick_interrupt

```
systick_interrupt_t enable_systick_interrupt
```

Whether or not the systick interrupt should be enabled. Recommended value si SYSTICK_INT_ENABLED.

4.1.2.4 tick_freq_khz

```
uint32_t tick_freq_khz
```

How quickly the systick should trigger in kHz. Recommended value is 1

The documentation for this struct was generated from the following file:

- /home/marko/Documents/embedded_workspace/systick_driver/systick_stm32f411_config.h

Chapter 5

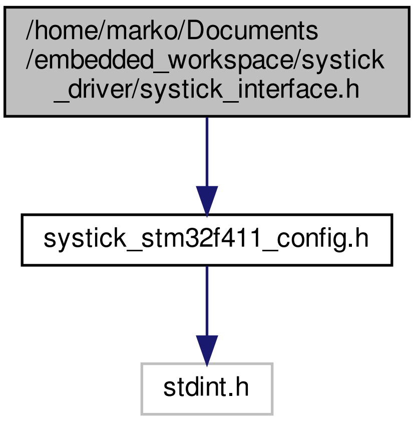
File Documentation

5.1 /home/marko/Documents/embedded_workspace/systick_driver/systick_interface.h File Reference

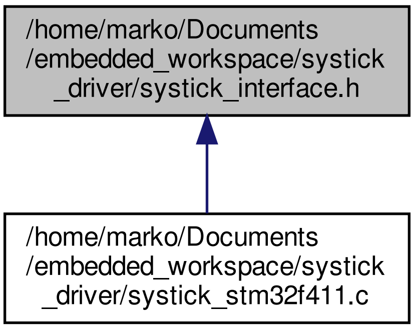
General interface covering user accesses to initialise the system wide tick for timeout/timekeeping operations.

```
#include "systick_stm32f411_config.h"
```

Include dependency graph for systick_interface.h:


/home/marko/Documents
/embedded_workspace/systick
_driver/systick_interface.h
systick_stm32f411_config.h
stdint.h

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


/home/marko/Documents/embedded_workspace/systick_driver/systick_interface.h
/home/marko/Documents/embedded_workspace/systick_driver/systick_stm32f411.c

Typedefs

- typedef void(∗ **systick_callback_t**) (void)

Functions

- void systick_init (systick_config_t ∗config)
- void systick_tick_freq_set (systick_config_t ∗config)
- void systick_interrupt_control (systick_interrupt_t interrupt_control)
- void systick_pause (void)
- void systick_resume (void)
- uint32_t systick_get_tick (void)
- void systick_delay (uint32_t delay_ms)
- void systick_increment (void)
- void systick_callback_register (systick_callback_t callback_func)
- void systick_irq_handler (void)

5.1.1 Detailed Description

General interface covering user accesses to initialise the system wide tick for timeout/timekeeping operations.

5.1.2 Function Documentation

5.1.2.1 systick_callback_register()

```
void systick_callback_register (
            systick_callback_t callback_func )
```

Description:

```
Registers the callback function as the desired on-interrupt functionality.
```

PRE-CONDITION: None.

POST-CONDITION: the systick_callback function pointer variable now points to the desired function

Parameters

callback_func	a function pointer to a void (∗function)(void)

Returns

void

Example:

```
systick_callback_register(&interrupt_behaviour);

//the irq handler will now call interrupt_behaviour
  SysTick_IRQHandler(void)
  {
      systick_irq_handler();
  }
```

See also

systick_irq_handler

- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.1.2.2 systick_delay()

```
void systick_delay (
              uint32_t delay_ms )
```

Description:

```
Delays the program for the duration of delay_ms in milliseconds
```

PRE-CONDITION: None

POST-CONDITION: delay_ms have gone by and the rest of the program will resume

Parameters

delay_ms	is the length of time the user wishes to way

Returns

void

Example:

```
systick_delay(200);
```

See also

systick_tick_freq_set
systick_get_tick

- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.1.2.3 systick_get_tick()

```
uint32_t systick_get_tick (
            void  )
```

Description:

```
DReturns the current value of the tick_ms variable
```

PRE-CONDITION: None

POST-CONDITION: The function has returned the current value of the tick variable.

Returns

uint32_t the current tick value

Example:

```
uint32_t current_tick = systick_get_tick();
```

See also

systick_tick_freq_set
systick_delay

- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.1.2.4 systick_increment()

```
void systick_increment (
             void  )
```

Description:

```
Increments the tick by the number of milliseconds between systick register
overflows. Called within systick_irq_handler.
```

PRE-CONDITION: None.

POST-CONDITION: tick_ms has incremented by tick_freq milliseconds

Returns

void

Example:

```
//By default is called automatically upon SysTick interrupt
  SysTick_IRQHandler(void)
  {
      systick_irq_handler();
  }
```

See also

systick_callback_register
systick_irq_handler

- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.1.2.5 systick_init()

```
void systick_init (
            systick_config_t * config )
```

Description:

```
Carries out the initialisation of the the systick based on information in the
config table
```

PRE-CONDITION: The clock system (RCC) has been initialised. PRE-CONDITION: The desired frequency (tick↩_freq_khz) results in a number small enough to fit the 0xFFFFFF mask PRE-CONDITION: (Soft Assert) the systick is enabled through its config register

POST-CONDITION: The systick has been configured to count with the desired frequency POST-CONDITION: The systick interrupt has been enabled (if desired) and its priority set to maximum. POST-CONDITION: The systick clock source has been set to the desired option

Parameters

config	a pointer to the systick configuration structure

Returns

void

Example:

```
systick_config_t *tick_config = systick_config_get();
systick_init(tick_config);
```

See also

systick_config_get
systick_tick_freq_set
systick_pause
systick_resume
systick_interrupt_control
- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.1.2.6 systick_interrupt_control()

```
void systick_interrupt_control (
            systick_interrupt_t interrupt_control )
```

Description:

```
Enables or disables the systick interrupt
```

PRE-CONDITION: (Soft Assert) The systick is paused

POST-CONDITION: The systick interrupt is enabled or disabled, as per the input

Parameters

systick_↩ interrupt_t	Control parameter defining if the interrupt will be activated or deactivated

Returns

void

Example:

```
systick_pause();
systick_interrupt_control(SYSTICK_INT_ENABLED);
systick_resume();
```

See also

systick_init
systick_tick_freq_set
systick_pause
systick_resume

- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.1.2.7 systick_irq_handler()

```
void systick_irq_handler (
            void )
```

Description:

```
Calls the systick callback function. The default callback is systick_increment.
```

PRE-CONDITION: The callback function is non-NULL

POST-CONDITION: the systick_callback function is called

Returns

void

Example:

```
systick_callback_register(&interrupt_behaviour);

//the irq handler will now call interrupt_behaviour
  SysTick_IRQHandler(void)
  {
      systick_irq_handler();
  }
```

See also

systick_callback_register
systick_increment

- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.1.2.8 systick_pause()

```
void systick_pause (
               void   )
```

Description:

```
Pauses the counting of the systick.
```

PRE-CONDITION: None

POST-CONDITION: The systick timer is paused

Returns

void

Example:

```
systick_pause();
//... do things....
systick_resume();
```

See also

systick_tick_freq_set
systick_resume

- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.1.2.9 systick_resume()

```
void systick_resume (
            void  )
```

Description:

```
Resume the counting of the systick.
```

PRE-CONDITION: None

POST-CONDITION: The systick timer is running

Returns

void

Example:

```
systick_pause();
//... do things....
systick_resume();
```

See also

systick_tick_freq_set
systick_pause

- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.1.2.10 systick_tick_freq_set()

```
void systick_tick_freq_set (
            systick_config_t * config )
```

Description:

```
Sets the frequency of the systick update to the desired value in kHz.
```

PRE-CONDITION: The desired frequency (tick_freq_khz) results in a number small enough to fit the 0xFFFFFF mask PRE-CONDITION: (Soft Assert) the systick is enabled through its config register PRE-CONDITION: (Soft Assert) the systick is paused

POST-CONDITION: The systick has been configured to count with the desired frequency

Parameters

config	a pointer to the systick configuration structure

Returns

void

Example:

```
systick_config_t *tick_config = systick_config_get();
systick_init(tick_config);
//... later ...
systick_pause();
tick_config->tick_freq_khz = 5; //kHz
systick_tick_freq_set(tick_config);
systick_resume();
```

See also

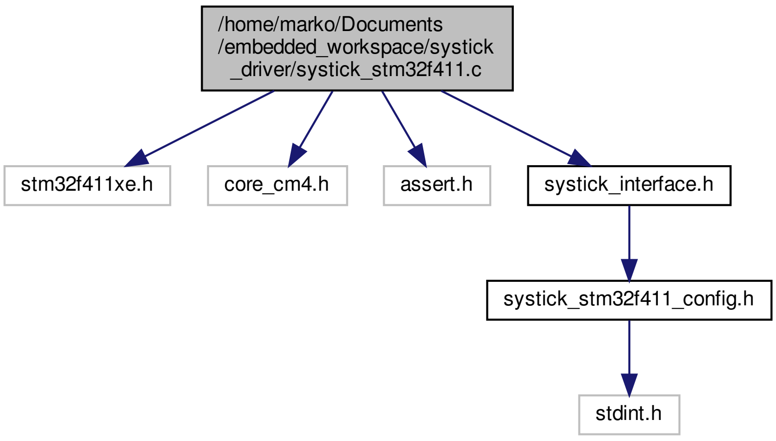
systick_init
systick_config_get
systick_pause
systick_resume
- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.2 /home/marko/Documents/embedded_workspace/systick_driver/systick_stm32f411.c File Reference

Chip specific implementation of systick control. Many functions are pointers to vendor created routines due to the very specific nature of system clocks and ticks.

```
#include "stm32f411xe.h"
#include "core_cm4.h"
#include <assert.h>
#include "systick_interface.h"
```
Include dependency graph for systick_stm32f411.c:


/home/marko/Documents
/embedded_workspace/systick
_driver/systick_stm32f411.c
stm32f411xe.h
core_cm4.h
assert.h
systick_interface.h
systick_stm32f411_config.h
stdint.h

Macros

- #define NULL (void ∗) 0

Functions

- void systick_init (systick_config_t ∗config)
- void systick_tick_freq_set (systick_config_t ∗config)
- void systick_pause (void)
- void systick_resume (void)
- void systick_interrupt_control (systick_interrupt_t interrupt_control)
- uint32_t systick_get_tick (void)
- void systick_delay (uint32_t delay_ms)
- void systick_increment (void)
- void systick_callback_register (systick_callback_t callback_func)
- void systick_irq_handler (void)

Variables

- static volatile uint32_t tick_ms = 0
- static uint32_t tick_freq
- static systick_callback_t systick_callback = systick_increment

5.2.1 Detailed Description

Chip specific implementation of systick control. Many functions are pointers to vendor created routines due to the very specific nature of system clocks and ticks.

Note

This implementation depends on CMSIS (core_cm4.h)

5.2.2 Macro Definition Documentation

5.2.2.1 NULL

```
#define NULL (void *) 0
```

Definition of NULL in case it is not defined elsewhere

5.2.3 Function Documentation

5.2.3.1 systick_callback_register()

```
void systick_callback_register (
            systick_callback_t callback_func )
```

Description:

```
Registers the callback function as the desired on-interrupt functionality.
```

PRE-CONDITION: None.

POST-CONDITION: the systick_callback function pointer variable now points to the desired function

Parameters

callback_func	a function pointer to a void (*function)(void)

Returns

void

Example:

```
systick_callback_register(&interrupt_behaviour);

//the irq handler will now call interrupt_behaviour
  SysTick_IRQHandler(void)
  {
      systick_irq_handler();
  }
```

See also

systick_irq_handler

- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.2.3.2 systick_delay()

```
void systick_delay (
              uint32_t delay_ms )
```

Description:

```
Delays the program for the duration of delay_ms in milliseconds
```

PRE-CONDITION: None

POST-CONDITION: delay_ms have gone by and the rest of the program will resume

Parameters

delay_ms	is the length of time the user wishes to way

Returns

void

Example:

```
systick_delay(200);
```

See also

systick_tick_freq_set
systick_get_tick

- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.2.3.3 systick_get_tick()

```
uint32_t systick_get_tick (
            void  )
```

Description:

```
DReturns the current value of the tick_ms variable
```

PRE-CONDITION: None

POST-CONDITION: The function has returned the current value of the tick variable.

Returns

uint32_t the current tick value

Example:

```
uint32_t current_tick = systick_get_tick();
```

See also

systick_tick_freq_set
systick_delay

- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.2.3.4 systick_increment()

```
void systick_increment (
            void  )
```

Description:

```
Increments the tick by the number of milliseconds between systick register
overflows. Called within systick_irq_handler.
```

PRE-CONDITION: None.

POST-CONDITION: tick_ms has incremented by tick_freq milliseconds

Returns

void

Example:

```
//By default is called automatically upon SysTick interrupt
  SysTick_IRQHandler(void)
  {
      systick_irq_handler();
  }
```

See also

systick_callback_register
systick_irq_handler

- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.2.3.5 systick_init()

```
void systick_init (
            systick_config_t * config )
```

Description:

```
Carries out the initialisation of the the systick based on information in the
config table
```

PRE-CONDITION: The clock system (RCC) has been initialised. PRE-CONDITION: The desired frequency (tick↩_freq_khz) results in a number small enough to fit the 0xFFFFFF mask PRE-CONDITION: (Soft Assert) the systick is enabled through its config register

POST-CONDITION: The systick has been configured to count with the desired frequency POST-CONDITION: The systick interrupt has been enabled (if desired) and its priority set to maximum. POST-CONDITION: The systick clock source has been set to the desired option

Parameters

config	a pointer to the systick configuration structure

Returns

void

Example:

```
systick_config_t *tick_config = systick_config_get();
systick_init(tick_config);
```

See also

systick_config_get
systick_tick_freq_set
systick_pause
systick_resume
systick_interrupt_control
- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.2.3.6 systick_interrupt_control()

```
void systick_interrupt_control (
            systick_interrupt_t interrupt_control )
```

Description:

```
Enables or disables the systick interrupt
```

PRE-CONDITION: (Soft Assert) The systick is paused

POST-CONDITION: The systick interrupt is enabled or disabled, as per the input

Parameters

systick_↩ interrupt_t	Control parameter defining if the interrupt will be activated or deactivated

Returns

void

Example:

```
systick_pause();
systick_interrupt_control(SYSTICK_INT_ENABLED);
systick_resume();
```

See also

systick_init
systick_tick_freq_set
systick_pause
systick_resume

- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.2.3.7 systick_irq_handler()

```
void systick_irq_handler (
            void )
```

Description:

```
Calls the systick callback function. The default callback is systick_increment.
```

PRE-CONDITION: The callback function is non-NULL

POST-CONDITION: the systick_callback function is called

Returns

void

Example:

```
systick_callback_register(&interrupt_behaviour);

//the irq handler will now call interrupt_behaviour
  SysTick_IRQHandler(void)
  {
      systick_irq_handler();
  }
```

See also

systick_callback_register
systick_increment

- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.2.3.8 systick_pause()

```
void systick_pause (
            void  )
```

Description:

```
Pauses the counting of the systick.
```

PRE-CONDITION: None

POST-CONDITION: The systick timer is paused

Returns

void

Example:

```
systick_pause();
//... do things....
systick_resume();
```

See also

systick_tick_freq_set
systick_resume

- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.2.3.9 systick_resume()

```
void systick_resume (
             void  )
```

Description:

```
Resume the counting of the systick.
```

PRE-CONDITION: None

POST-CONDITION: The systick timer is running

Returns

void

Example:

```
systick_pause();
//... do things....
systick_resume();
```

See also

systick_tick_freq_set
systick_pause

- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.2.3.10 systick_tick_freq_set()

```
void systick_tick_freq_set (
             systick_config_t * config )
```

Description:

```
Sets the frequency of the systick update to the desired value in kHz.
```

PRE-CONDITION: The desired frequency (tick_freq_khz) results in a number small enough to fit the 0xFFFFFF mask PRE-CONDITION: (Soft Assert) the systick is enabled through its config register PRE-CONDITION: (Soft Assert) the systick is paused

POST-CONDITION: The systick has been configured to count with the desired frequency

Parameters

config	a pointer to the systick configuration structure

Returns

void

Example:

```
systick_config_t *tick_config = systick_config_get();
systick_init(tick_config);
//... later ...
systick_pause();
tick_config->tick_freq_khz = 5; //kHz
systick_tick_freq_set(tick_config);
systick_resume();
```

See also

systick_init
systick_config_get
systick_pause
systick_resume
- CHANGE HISTORY -

Date	Software Version	Initials	Description

5.2.4 Variable Documentation

5.2.4.1 systick_callback

```
systick_callback_t systick_callback = systick_increment  [static]
```

Callback function which will be dereferenced upon systick interrupts Default value is systick_increment, but can be changed through the callback_register function

5.2.4.2 tick_freq

```
uint32_t tick_freq  [static]
```

Tick frequency (increment rate)

5.2.4.3 tick_ms

```
volatile uint32_t tick_ms = 0  [static]
```

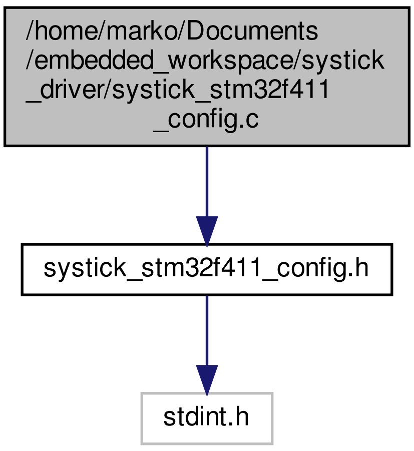
Encapsulated tick value

5.3 /home/marko/Documents/embedded_workspace/systick_driver/systick_stm32f411↩ _config.c File Reference

Contains the configuration information for the systick.

```
#include "systick_stm32f411_config.h"
```

Include dependency graph for systick_stm32f411_config.c:


/home/marko/Documents
/embedded_workspace/systick
_driver/systick_stm32f411
_config.c
systick_stm32f411_config.h
stdint.h

Functions

- const systick_config_t ∗ systick_config_get (void)

Variables

- static const systick_config_t systick_config_table [NUM_SYSTICKS]

5.3.1 Detailed Description

Contains the configuration information for the systick.

5.3.2 Function Documentation

5.3.2.1 systick_config_get()

```
const systick_config_t* systick_config_get (
            void  )
```

Function returning a pointer to the (quite protected) config data

5.3.3 Variable Documentation

5.3.3.1 systick_config_table

```
const systick_config_t systick_config_table[NUM_SYSTICKS]  [static]
```

Initial value:

```
=
{
        {SYSTICK_ENABLED,    1,         SYSTICK_INT_ENABLED,    SYSTICK_INTERNAL_CLOCK}
}
```

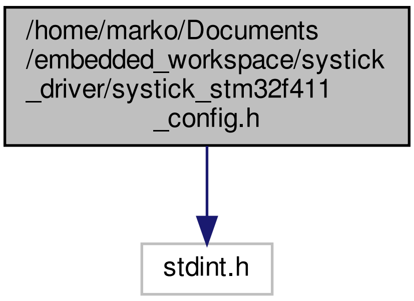
Table containing config information for the configuration of the systick. Populated at first with default values

5.4 /home/marko/Documents/embedded_workspace/systick_driver/systick_stm32f411↩_config.h File Reference

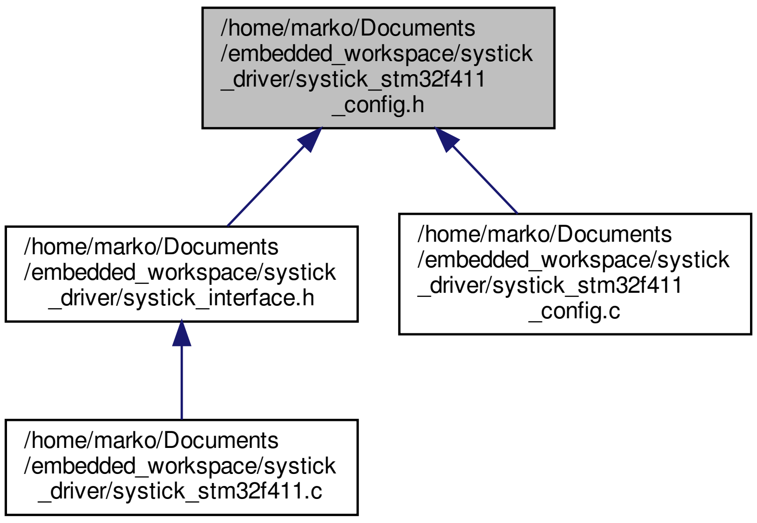
Chip specific header containing all relevant enums and structs to configure the systick.

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

Include dependency graph for systick_stm32f411_config.h:



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


/home/marko/Documents/embedded_workspace/systick_driver/systick_stm32f411_config.h
/home/marko/Documents/embedded_workspace/systick_driver/systick_interface.h
/home/marko/Documents/embedded_workspace/systick_driver/systick_stm32f411_config.c
/home/marko/Documents/embedded_workspace/systick_driver/systick_stm32f411.c

Data Structures

- struct systick_config_t

Enumerations

- enum **systick_t** { **SYSTICK_1**, **NUM_SYSTICKS** }
- enum systick_enabled_t { **SYSTICK_DISABLED**, **SYSTICK_ENABLED** }
- enum systick_interrupt_t { **SYSTICK_INT_DISABLED**, **SYSTICK_INT_ENABLED** }
- enum systick_clock_source_t { **SYSTICK_EXTERNAL_CLOCK**, **SYSTICK_INTERNAL_CLOCK** }

Functions

- const systick_config_t ∗ systick_config_get (void)

Variables

- uint32_t SystemCoreClock

5.4.1 Detailed Description

Chip specific header containing all relevant enums and structs to configure the systick.

5.4.2 Enumeration Type Documentation

5.4.2.1 systick_clock_source_t

```
enum systick_clock_source_t
```

Options for where the systick gets its clock. Internal clock is the default.

5.4.2.2 systick_enabled_t

```
enum systick_enabled_t
```

Contains options to enable or disable the systick. Note that a disabled systick will disable timeout features for all communication buses

5.4.2.3 systick_interrupt_t

```
enum systick_interrupt_t
```

Enables or disables the systick interrupt. The systick should be enabled to allow updating of the source-file scoped timer variable every x ms.

5.4.3 Function Documentation

5.4.3.1 systick_config_get()

```
const systick_config_t* systick_config_get (
             void  )
```

Function returning a pointer to the (quite protected) config data

5.4.4 Variable Documentation

5.4.4.1 SystemCoreClock

```
uint32_t SystemCoreClock
```

Core clock frequency as defined in system_stm32f4xx.c by STM