FreeRTOS CLI + STM32 by Aaron Escoboza

1.0

Generated by Doxygen 1.10.0

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

- Stm32f4xx_system

3.1.1 Detailed Description

3.1.2 Stm32f4xx_system

Topics

- STM32F4xx_System_Private_Includes
- STM32F4xx_System_Private_TypesDefinitions
- STM32F4xx_System_Private_Defines
- STM32F4xx_System_Private_Macros
- STM32F4xx_System_Private_Variables
- STM32F4xx_System_Private_FunctionPrototypes
- STM32F4xx_System_Private_Functions

3.1.2.1 Detailed Description

3.1.2.2 STM32F4xx_System_Private_Includes

Macros

- #define HSE_VALUE ((uint32_t)25000000)
- #define HSI_VALUE ((uint32_t)16000000)

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 ((uint32_t)25000000)
```

Default value of the External oscillator in Hz

3.1.2.2.2.2 HSI_VALUE

```
#define HSI_VALUE ((uint32_t)16000000)
```

Value of the Internal oscillator in Hz

3.1.2.3 STM32F4xx_System_Private_TypesDefinitions

3.1.2.4 STM32F4xx_System_Private_Defines

3.1.2.5 STM32F4xx_System_Private_Macros

3.1.2.6 STM32F4xx_System_Private_Variables

Variables

- uint32_t **SystemCoreClock** = 16000000
- 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}

3.1.2.6.1 Detailed Description

3.1.2.7 STM32F4xx_System_Private_FunctionPrototypes

3.1.2.8 STM32F4xx_System_Private_Functions

Functions

- void SystemInit (void)

  Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.
- 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 stm32f4xx_hal_conf.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 stm32f4xx_hal_conf.h file (its value depends on the application requirements), 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 Initialize the FPU setting, vector table location and External memory configuration.

Parameters

None	

Return values

None	

Chapter 4

File Documentation

4.1 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩ RTOS/Core/bsp/inc/bsp.h File Reference

Header file to expose a BSP generic functions.

Functions

- BspError_e bspInit (void)

  Calls all BSP init functions.

4.1.1 Detailed Description

Header file to expose a BSP generic functions.

Author

Aaron Escoboza, Github account: `https://github.com/aaron-ev`

4.1.2 Function Documentation

4.1.2.1 bspInit()

```
BspError_e bspInit (
             void  )
```

Calls all BSP init functions.

Parameters

void	

Return values

BspError←↩ _e	

4.2 bsp.h

Go to the documentation of this file.

```
#ifndef __BSP__H
#define __BSP__H

#include "bspPwm.h"
#include "bspGpio.h"
#include "bspClk.h"
#include "bspRtc.h"

BspError_e bspInit(void);

#endif
```

4.3 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree←↩ RTOS/Core/bsp/inc/bspClk.h File Reference

Header file for clock information.

Functions

- void bspGetClockIinfo (char ∗pcWriteBuffer, size_t xWriteBufferLen)

  Gets system clock, PCLKx and CLK dividers.

4.3.1 Detailed Description

Header file for clock information.

Author

Aaron Escoboza, Github account: `https://github.com/aaron-ev`

4.3.2 Function Documentation

4.3.2.1 bspGetClockIinfo()

```
void bspGetClockIinfo (
            char ∗ pcWriteBuffer,
            size_t xWriteBufferLen )
```

Gets system clock, PCLKx and CLK dividers.

Parameters

∗pcWriteBuffer	pointer to buffer where clock information will be stored.
xWriteBufferLen	buffer length.

Return values

void	

4.4 bspClk.h

Go to the documentation of this file.

```
#ifndef __BSP_CLK_H
#define __BSP_CLK_H


#include "stdio.h"
#include "stdint.h"
#include "stm32f4xx_hal.h"

void bspGetClockIinfo(char *pcWriteBuffer, size_t xWriteBufferLen);

#endif
```

4.5 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↵ RTOS/Core/bsp/inc/bspGpio.h File Reference

Header file that exposes GPIO data types and GPIo APIs.

Enumerations

- enum **BspGpioInstance_e** {
  BSP_GPIOA , **BSP_GPIOB** , **BSP_GPIOC** , **BSP_GPIOD** ,
  BSP_GPIOE , **BSP_GPIOH** , **BSP_MAX_GPIO_INSTANCE** }
- enum **BspPinNum_e** {
  BSP_GPIO_PIN_0 , **BSP_GPIO_PIN_1** , **BSP_GPIO_PIN_2** , **BSP_GPIO_PIN_3** ,
  BSP_GPIO_PIN_4 , **BSP_GPIO_PIN_5** , **BSP_GPIO_PIN_6** , **BSP_GPIO_PIN_7** ,
  BSP_GPIO_PIN_8 , **BSP_GPIO_PIN_9** , **BSP_GPIO_PIN_10** , **BSP_GPIO_PIN_11** ,
  BSP_GPIO_PIN_12 , **BSP_GPIO_PIN_13** , **BSP_GPIO_PIN_14** , **BSP_GPIO_PIN_15** }
- enum **BspGpioPinState_e** { **BSP_GPIO_PIN_LOW** , **BSP_GPIO_PIN_HIGH** }

Functions

- void bspGpioToggle (BspGpioInstance_e eGpio, BspPinNum_e pinNum)

  Toggles a GPIO pin.
- BspGpioInstance_e bspGpioMapInstance (const char pcGpioInstance)

  Maps a letter (a, b, c, d, e , h) to a BSP GPIO instance.
- BspGpioPinState_e bspGpioRead (BspGpioInstance_e eGpio, BspPinNum_e pinNum)

  Reads from a GPIO pin.
- void bspGpioWrite (BspGpioInstance_e eGpio, BspPinNum_e pinNum, BspGpioPinState_e pinState)

  Writes to a GPIO pin.

4.5.1 Detailed Description

Header file that exposes GPIO data types and GPIo APIs.

Author

Aaron Escoboza, Github account: `https://github.com/aaron-ev`

4.5.2 Function Documentation

4.5.2.1 bspGpioMapInstance()

```
BspGpioInstance_e bspGpioMapInstance (
             const char pcGpioInstance )
```

Maps a letter (a, b, c, d, e , h) to a BSP GPIO instance.

Parameters

pcGpioInstance	GPIO instance character.

Return values

BSP	GPIO instance.

4.5.2.2 bspGpioRead()

```
BspGpioPinState_e bspGpioRead (
             BspGpioInstance_e eGpio,
             BspPinNum_e pinNum )
```

Reads from a GPIO pin.

Parameters

eGpio	BSP GPIO instance.
pinNum	BSP GPIO pin number.

Return values

BSP	pin state

4.5.2.3 bspGpioToggle()

```
void bspGpioToggle (
             BspGpioInstance_e eGpio,
             BspPinNum_e pinNum )
```

Toggles a GPIO pin.

Parameters

eGpio	BSP GPIO instance.
pinNum	BSP GPIO pin number.

Return values

void	

4.5.2.4 bspGpioWrite()

```
void bspGpioWrite (
              BspGpioInstance_e eGpio,
              BspPinNum_e pinNum,
              BspGpioPinState_e pinState )
```

Writes to a GPIO pin.

Parameters

eGpio	BSP GPIO instance.
pinNum	BSP GPIO pin number.
pinState	new BSP pin state.

Return values

void	

4.6 bspGpio.h

Go to the documentation of this file.

```

#ifndef __BSP_GPIO_H
#define __BSP_GPIO_H

#include "stdint.h"
#include "stm32f4xx_hal.h"

typedef enum
{
    BSP_GPIOA,
    BSP_GPIOB,
    BSP_GPIOC,
    BSP_GPIOD,
    BSP_GPIOE,
    BSP_GPIOH,
    BSP_MAX_GPIO_INSTANCE,
}BspGpioInstance_e;

typedef enum
{
    BSP_GPIO_PIN_0,
    BSP_GPIO_PIN_1,
    BSP_GPIO_PIN_2,
    BSP_GPIO_PIN_3,
    BSP_GPIO_PIN_4,
    BSP_GPIO_PIN_5,
    BSP_GPIO_PIN_6,
    BSP_GPIO_PIN_7,
    BSP_GPIO_PIN_8,
    BSP_GPIO_PIN_9,
    BSP_GPIO_PIN_10,
    BSP_GPIO_PIN_11,
    BSP_GPIO_PIN_12,
    BSP_GPIO_PIN_13,
    BSP_GPIO_PIN_14,
    BSP_GPIO_PIN_15,
}BspPinNum_e;

typedef enum
{
    BSP_GPIO_PIN_LOW,
    BSP_GPIO_PIN_HIGH,
}BspGpioPinState_e;

void bspGpioToggle(BspGpioInstance_e eGpio, BspPinNum_e pinNum);
BspGpioInstance_e bspGpioMapInstance(const char pcGpioInstance);
BspGpioPinState_e bspGpioRead(BspGpioInstance_e eGpio, BspPinNum_e pinNum);
void bspGpioWrite(BspGpioInstance_e eGpio, BspPinNum_e pinNum, BspGpioPinState_e pinState);

#endif
```

4.7 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩ RTOS/Core/bsp/inc/bspPwm.h File Reference

Header file that exposes PWM data types and PWM APIs.

Enumerations

- enum **pwmChannels_e** {
  PWM_CH_1 , **PWM_CH_2** , **PWM_CH_3** , **PWM_CH_4** ,
  MAX_PWM_CH }

Functions

- BspError_e bspPwmInit (void)

  Initialize the timer init and the PWM channel.
- TIM_HandleTypeDef ∗ bspPwmGetHandler (void)

  Gets the timer handler associated to a PWM channel.
- BspError_e bspPwmSetFreq (uint32_t uNewFreq)

  Sets a new frequency.
- void bspPwmStart (pwmChannels_e eChannelIndex)

  Starts a PWM cannel.
- BspError_e bspPwmSetDuty (uint8_t uNewDuty, pwmChannels_e xChannel)

  Sets a new duty cycle to a giving channel.

4.7.1 Detailed Description

Header file that exposes PWM data types and PWM APIs.

Author

Aaron Escoboza, Github account: `https://github.com/aaron-ev`

4.7.2 Function Documentation

4.7.2.1 bspPwmGetHandler()

```
TIM_HandleTypeDef * bspPwmGetHandler (
            void  )
```

Gets the timer handler associated to a PWM channel.

Parameters

void	

Return values

Pointer	to the timer handler.

4.7.2.2 bspPwmInit()

```
BspError_e bspPwmInit (
            void  )
```

Initialize the timer init and the PWM channel.

Parameters

void	

Return values

BSP	status

4.7.2.3 bspPwmSetDuty()

```
BspError_e bspPwmSetDuty (
            uint8_t uNewDuty,
            pwmChannels_e xChannel )
```

Sets a new duty cycle to a giving channel.

Parameters

uNewDuty	Duty cycle to be set
xChannel	PWM channel

Return values

HAL	status

4.7.2.4 bspPwmSetFreq()

```
BspError_e bspPwmSetFreq (
             uint32_t uNewFreq )
```

Sets a new frequency.

Parameters

uNewFreq	Frequency to be set

Return values

BSP	status

Note

1 decimal value = 1Hz

4.7.2.5 bspPwmStart()

```
void bspPwmStart (
             pwmChannels_e eChannelIndex )
```

Starts a PWM cannel.

Parameters

eChannelIdex	BSP channel number

Return values

void	

4.8 bspPwm.h

Go to the documentation of this file.

```

#ifndef __BSP_PWM_H
#define __BSP_PWM_H

#include "stdint.h"
#include "stm32f4xx_hal.h"
#include "bspTypeDef.h"

typedef enum
{
    PWM_CH_1,
    PWM_CH_2,
    PWM_CH_3,
    PWM_CH_4,
    MAX_PWM_CH,
} pwmChannels_e;
```

```

BspError_e bspPwmInit(void);
TIM_HandleTypeDef* bspPwmGetHandler(void);
BspError_e bspPwmSetFreq(uint32_t uNewFreq);
void bspPwmStart(pwmChannels_e eChannelIndex);
BspError_e bspPwmSetDuty(uint8_t uNewDuty, pwmChannels_e xChannel);

#endif
```

4.9 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩ RTOS/Core/bsp/inc/bspRtc.h File Reference

Header file that exposes RTC APIs.

Functions

- BspError_e bspRtcInit (void)

  Initialize RTC peripheral.
- BspError_e bspRtcGetTime (BspRtcTime ∗bspRtcTime)

  Get the current time stored in RTC registers.
- BspError_e bspRtcSetTime (BspRtcTime ∗bspRtcTime)

  Set a new time to RTC registers.

4.9.1 Detailed Description

Header file that exposes RTC APIs.

Author

Aaron Escoboza, Github account: `https://github.com/aaron-ev`

4.9.2 Function Documentation

4.9.2.1 bspRtcGetTime()

```
BspError_e bspRtcGetTime (
            BspRtcTime ∗ bspRtcTime )
```

Get the current time stored in RTC registers.

Parameters

bspRtcTime	pointer to a RTC timer structure

Return values

BSP	error

4.9.2.2 bspRtcInit()

```
BspError_e bspRtcInit (
             void )
```

Initialize RTC peripheral.

Parameters

void	

Return values

BSP	error

4.9.2.3 bspRtcSetTime()

```
BspError_e bspRtcSetTime (
             BspRtcTime * bspRtcTime )
```

Set a new time to RTC registers.

Parameters

bspRtcTime	pointer to a RTC timer structure

Return values

BSP	error

4.10 bspRtc.h

Go to the documentation of this file.

```

#ifndef __BSP_RTC_H
#define __BSP_RTC_H

#include "stdint.h"
#include "stm32f4xx_hal.h"
#include "bspTypeDef.h"

typedef struct
{
    uint8_t uHours;
    uint8_t uMinutes;
    uint8_t uSeconds;
}BspRtcTime;

BspError_e bspRtcInit(void);
BspError_e bspRtcGetTime(BspRtcTime* bspRtcTime);
BspError_e bspRtcSetTime(BspRtcTime* bspRtcTime);

#endif
```

4.11 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩ RTOS/Core/bsp/inc/bspTypeDef.h File Reference

Header that contains BSP definitions.

Enumerations

- enum **BspError_e** { **BSP_NO_ERROR** , **BSP_ERROR_EIO** = EIO , **BSP_ERROR_EINVAL** = EINVAL }

4.11.1 Detailed Description

Header that contains BSP definitions.

Author

Aaron Escoboza, Github account: https://github.com/aaron-ev

4.12 bspTypeDef.h

Go to the documentation of this file.

```
#ifndef  __BSP_TYPE_DEF_H
#define __BSP_TYPE_DEF_H

#include "errno.h"

typedef enum
{
    BSP_NO_ERROR,
    BSP_ERROR_EIO = EIO,
    BSP_ERROR_EINVAL = EINVAL,
} BspError_e;

#endif
```

4.13 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩ RTOS/Core/bsp/src/bsp.c File Reference

source file to implement low level initializations.

Functions

- BspError_e bspConfigureTimForRunTimeStats (void)

  Configure timer used for FreeRTOS task statistics.
- uint32_t bspGetTimStatsCount (void)

  Get current timer counter for FreeRTOS task statistics.
- BspError_e bspConsoleInit (void)

  Initialize UART frame.
- BspError_e bspInit (void)

  Calls all BSP init functions.

Variables

- UART_HandleTypeDef **consoleHandle**
- TIM_HandleTypeDef **xTimStatsHandler**

4.13.1 Detailed Description

source file to implement low level initializations.

Author

Aaron Escoboza, Github account: https://github.com/aaron-ev

4.13.2 Function Documentation

4.13.2.1 bspConfigureTimForRunTimeStats()

```
BspError_e bspConfigureTimForRunTimeStats (
            void  )
```

Configure timer used for FreeRTOS task statistics.

Parameters

void	

Return values

BSP	error

4.13.2.2 bspConsoleInit()

```
BspError_e bspConsoleInit (
            void  )
```

Initialize UART frame.

Parameters

void	

Return values

BSP	error

4.13.2.3 bspGetTimStatsCount()

```
uint32_t bspGetTimStatsCount (
             void  )
```

Get current timer counter for FreeRTOS task statistics.

Parameters

void	

Return values

void	

4.13.2.4 bspInit()

```
BspError_e bspInit (
             void  )
```

Calls all BSP init functions.

Parameters

void	

Return values

BspError↩ _e	

4.14 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩ RTOS/Core/bsp/src/bspClk.c File Reference

source file to implement functions related to the clock tree.

Functions

- void bspGetClockIinfo (char ∗pcWriteBuffer, size_t xWriteBufferLen)
  Gets system clock, PCLKx and CLK dividers.

4.14.1 Detailed Description

source file to implement functions related to the clock tree.

Author

Aaron Escoboza, Github account: `https://github.com/aaron-ev`

4.14.2 Function Documentation

4.14.2.1 bspGetClocklinfo()

```
void bspGetClockIinfo (
             char * pcWriteBuffer,
             size_t xWriteBufferLen )
```

Gets system clock, PCLKx and CLK dividers.

Parameters

∗pcWriteBuffer	pointer to buffer where clock information will be stored.
xWriteBufferLen	buffer length.

Return values

void	

4.15 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩RTOS/Core/bsp/src/bspGpio.c File Reference

source file to implement functions related to GPIO operations.

Functions

- BspGpioInstance_e bspGpioMapInstance (const char pcGpioInstance)
  Maps a letter (a, b, c, d, e , h) to a BSP GPIO instance.
- void bspGpioToggle (BspGpioInstance_e eGpio, BspPinNum_e pinNum)
  Toggles a GPIO pin.
- void bspGpioWrite (BspGpioInstance_e eGpio, BspPinNum_e pinNum, BspGpioPinState_e pinState)
  Writes to a GPIO pin.
- BspGpioPinState_e bspGpioRead (BspGpioInstance_e eGpio, BspPinNum_e pinNum)
  Reads from a GPIO pin.

4.15.1 Detailed Description

source file to implement functions related to GPIO operations.

Author

Aaron Escoboza, Github account: `https://github.com/aaron-ev`

4.15.2 Function Documentation

4.15.2.1 bspGpioMapInstance()

```
BspGpioInstance_e bspGpioMapInstance (
             const char pcGpioInstance )
```

Maps a letter (a, b, c, d, e , h) to a BSP GPIO instance.

Parameters

pcGpioInstance	GPIO instance character.

Return values

BSP	GPIO instance.

4.15.2.2 bspGpioRead()

```
BspGpioPinState_e bspGpioRead (
             BspGpioInstance_e eGpio,
             BspPinNum_e pinNum )
```

Reads from a GPIO pin.

Parameters

eGpio	BSP GPIO instance.
pinNum	BSP GPIO pin number.

Return values

BSP	pin state

4.15.2.3 bspGpioToggle()

```
void bspGpioToggle (
             BspGpioInstance_e eGpio,
             BspPinNum_e pinNum )
```

Toggles a GPIO pin.

Parameters

eGpio	BSP GPIO instance.
pinNum	BSP GPIO pin number.

Return values

void	

4.15.2.4 bspGpioWrite()

```
void bspGpioWrite (
             BspGpioInstance_e eGpio,
             BspPinNum_e pinNum,
             BspGpioPinState_e pinState )
```

Writes to a GPIO pin.

Parameters

eGpio	BSP GPIO instance.
pinNum	BSP GPIO pin number.
pinState	new BSP pin state.

Return values

void	

4.16 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩ RTOS/Core/bsp/src/bspPwm.c File Reference

source file to implement functions related to PWM signal manipulation.

Macros

- #define **PWM_MAX_CHANNELS** 4
- #define **PWM_DEFAULT_FREQ** 1000 /∗ 1kHz when clk is 1Mhz ∗/

Functions

- TIM_HandleTypeDef ∗ bspPwmGetHandler (void)

  Gets the timer handler associated to a PWM channel.
- void bspPwmStart (pwmChannels_e eChannelIndex)

  Starts a PWM cannel.
- BspError_e bspPwmSetFreq (uint32_t uNewFreq)

  Sets a new frequency.
- BspError_e bspPwmSetDuty (uint8_t uNewDuty, pwmChannels_e xChannel)

  Sets a new duty cycle to a giving channel.
- BspError_e bspPwmInit (void)

  Initialize the timer init and the PWM channel.

Variables

- PwmConfigStruct **pwmConfigStruct**

4.16.1 Detailed Description

source file to implement functions related to PWM signal manipulation.

Author

Aaron Escoboza, Github account: `https://github.com/aaron-ev`

4.16.2 Function Documentation

4.16.2.1 bspPwmGetHandler()

```
TIM_HandleTypeDef * bspPwmGetHandler (
             void  )
```

Gets the timer handler associated to a PWM channel.

Parameters

void	

Return values

Pointer	to the timer handler.

4.16.2.2 bspPwmInit()

```
BspError_e bspPwmInit (
             void  )
```

Initialize the timer init and the PWM channel.

Parameters

void	

Return values

BSP	status

4.16.2.3 bspPwmSetDuty()

```
BspError_e bspPwmSetDuty (
             uint8_t uNewDuty,
             pwmChannels_e xChannel )
```

Sets a new duty cycle to a giving channel.

Parameters

uNewDuty	Duty cycle to be set
xChannel	PWM channel

Return values

HAL	status

4.16.2.4 bspPwmSetFreq()

```
BspError_e bspPwmSetFreq (
            uint32_t uNewFreq )
```

Sets a new frequency.

Parameters

uNewFreq	Frequency to be set

Return values

BSP	status

Note

1 decimal value = 1Hz

4.16.2.5 bspPwmStart()

```
void bspPwmStart (
            pwmChannels_e eChannelIndex )
```

Starts a PWM cannel.

Parameters

eChannelIdex	BSP channel number

Return values

void	

4.17 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩ RTOS/Core/Inc/appConfig.h File Reference

Hold general application configuration.

Macros

- #define **DEBUG_PRINT_EN** 1 /∗ 1 = Enable , 0 = Disable ∗/
- #define **HEART_BEAT_PRIORITY_TASK** 1
- #define **HEART_BEAT_LED_PORT** GPIOC
- #define **HEART_BEAT_LED_PIN** GPIO_PIN_13

- #define **HEART_BEAT_BLINK_DELAY** 500 /∗ In ms ∗/
- #define **CONSOLE_INSTANCE** USART1
- #define **CONSOLE_TX_PIN** GPIO_PIN_6
- #define **CONSOLE_RX_PIN** GPIO_PIN_7
- #define **CONSOLE_GPIO_PORT** GPIOB
- #define **CONSOLE_BAUDRATE** 9600
- #define **CONSOLE_TASK_PRIORITY** 1
- #define **CONSOLE_STACK_SIZE** 3000
- #define **PWM_GPIO_INSTANCE** GPIOA
- #define **PWM_GPIO_PINX** GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3
- #define **PWM_GPIO_ALTERNATE** GPIO_AF1_TIM2
- #define **PWM_TIM_INSTANCE** TIM2

4.17.1 Detailed Description

Hold general application configuration.

Author

Aaron Escoboza, Github account: https://github.com/aaron-ev

4.18 appConfig.h

Go to the documentation of this file.

```

#ifndef APP_CONFIG__H
#define APP_CONFIG__H

/* Enable debug messages */
#define DEBUG_PRINT_EN                    1 /* 1 = Enable , 0 =  Disable */

/* Task priorities */
#define HEART_BEAT_PRIORITY_TASK          1

/* Heart beat settings */
#define HEART_BEAT_LED_PORT               GPIOC
#define HEART_BEAT_LED_PIN                GPIO_PIN_13
#define HEART_BEAT_BLINK_DELAY            500 /* In ms */

/* CLI console settings */
#define CONSOLE_INSTANCE                  USART1
#define CONSOLE_TX_PIN                    GPIO_PIN_6
#define CONSOLE_RX_PIN                    GPIO_PIN_7
#define CONSOLE_GPIO_PORT                 GPIOB
#define CONSOLE_BAUDRATE                  9600
#define CONSOLE_TASK_PRIORITY             1
#define CONSOLE_STACK_SIZE                3000

/* PWM signal settings */
#define PWM_GPIO_INSTANCE                 GPIOA
#define PWM_GPIO_PINX                     GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3
#define PWM_GPIO_ALTERNATE                GPIO_AF1_TIM2
#define PWM_TIM_INSTANCE                  TIM2

#endif
```

4.19 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩RTOS/Core/Inc/console.h File Reference

Console header file: APIs to handle the console.

Functions

- BaseType_t xbspConsoleInit (uint16_t usStackSize, UBaseType_t uxPriority, UART_HandleTypeDef ∗pxUartHandle)

  Initialize the console by registering all commands and creating a task.

4.19.1 Detailed Description

Console header file: APIs to handle the console.

Author

Aaron Escoboza, Github account: `https://github.com/aaron-ev`

4.19.2 Function Documentation

4.19.2.1 xbspConsoleInit()

```
BaseType_t xbspConsoleInit (
            uint16_t usStackSize,
            UBaseType_t uxPriority,
            UART_HandleTypeDef * pxUartHandle )
```

Initialize the console by registering all commands and creating a task.

Parameters

usStackSize	Task console stack size
uxPriority	Task console priority
∗pxUartHandle	Pointer for uart handle.

Return values

FreeRTOS	status

4.20 console.h

Go to the documentation of this file.

```

#ifndef __CONSOLE__H
#define __CONSOLE__H

#include "FreeRTOS.h"

BaseType_t xbspConsoleInit(uint16_t usStackSize, UBaseType_t uxPriority, UART_HandleTypeDef
      *pxUartHandle);

#endif
```

4.21 FreeRTOSConfig.h

```
/*
 * FreeRTOS V202112.00
 * Copyright (C) 2020 Amazon.com, Inc. or its affiliates.  All Rights Reserved.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 * this software and associated documentation files (the "Software"), to deal in
 * the Software without restriction, including without limitation the rights to
 * use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
 * the Software, and to permit persons to whom the Software is furnished to do so,
 * subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in all
 * copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
 * FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 * COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
 * IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 *
 * http://www.FreeRTOS.org
 * http://aws.amazon.com/freertos
 *
 * 1 tab == 4 spaces!
 */

#ifndef FREERTOS_CONFIG_H
#define FREERTOS_CONFIG_H

/*-----------------------------------------------------------
 * Application specific definitions.
 *
 * These definitions should be adjusted for your particular hardware and
 * application requirements.
 *
 * THESE PARAMETERS ARE DESCRIBED WITHIN THE 'CONFIGURATION' SECTION OF THE
 * FreeRTOS API DOCUMENTATION AVAILABLE ON THE FreeRTOS.org WEB SITE.
 *
 * See http://www.freertos.org/a00110.html
 *----------------------------------------------------------*/

/* Ensure stdint is only used by the compiler, and not the assembler. */
#if defined(__ICCARM__) || defined(__GNUC__) || defined(__CC_ARM)
    #include <stdint.h>
    extern uint32_t SystemCoreClock;
#endif

#define configUSE_PREEMPTION 1
#define configUSE_IDLE_HOOK 0
#define configUSE_TICK_HOOK 0
#define configCPU_CLOCK_HZ (SystemCoreClock)
#define configTICK_RATE_HZ ((TickType_t)1000)
#define configMAX_PRIORITIES (5)
#define configMINIMAL_STACK_SIZE ((unsigned short)130)
//#define configTOTAL_HEAP_SIZE ((size_t)(75 * 1024))
#define configTOTAL_HEAP_SIZE ((size_t)(75 * 524))
#define configMAX_TASK_NAME_LEN (10)
#define configUSE_TRACE_FACILITY 1
#define configUSE_16_BIT_TICKS 0
#define configIDLE_SHOULD_YIELD 1
#define configUSE_MUTEXES 1
#define configQUEUE_REGISTRY_SIZE 8
#define configCHECK_FOR_STACK_OVERFLOW 0
#define configUSE_RECURSIVE_MUTEXES 1
#define configUSE_MALLOC_FAILED_HOOK 0
#define configUSE_APPLICATION_TASK_TAG 0
#define configUSE_COUNTING_SEMAPHORES 1
#define configGENERATE_RUN_TIME_STATS 1
#define configUSE_TASK_NOTIFICATIONS 1
#define configTASK_NOTIFICATION_ARRAY_ENTRIES 2
#define configCOMMAND_INT_MAX_OUTPUT_SIZE 500

/* Co-routine definitions. */
#define configUSE_CO_ROUTINES 0
#define configMAX_CO_ROUTINE_PRIORITIES (2)

/* Software timer definitions. */
#define configUSE_TIMERS 1
#define configTIMER_TASK_PRIORITY (2)
#define configTIMER_QUEUE_LENGTH 10
#define configTIMER_TASK_STACK_DEPTH (configMINIMAL_STACK_SIZE * 2)

/* Set the following definitions to 1 to include the API function, or zero
to exclude the API function. */
```

```
#define INCLUDE_vTaskPrioritySet 1
#define INCLUDE_uxTaskPriorityGet 1
#define INCLUDE_vTaskDelete 1
#define INCLUDE_vTaskCleanUpResources 1
#define INCLUDE_vTaskSuspend 1
#define INCLUDE_vTaskDelayUntil 1
#define INCLUDE_vTaskDelay 1

/* Cortex-M specific definitions. */
#ifdef __NVIC_PRIO_BITS
/* __BVIC_PRIO_BITS will be specified when CMSIS is being used. */
#define configPRIO_BITS __NVIC_PRIO_BITS
#else
#define configPRIO_BITS 4 /* 15 priority levels */
#endif

/* The lowest interrupt priority that can be used in a call to a "set priority"
function. */
#define configLIBRARY_LOWEST_INTERRUPT_PRIORITY 0xf

/* The highest interrupt priority that can be used by any interrupt service
routine that makes calls to interrupt safe FreeRTOS API functions.  DO NOT CALL
INTERRUPT SAFE FREERTOS API FUNCTIONS FROM ANY INTERRUPT THAT HAS A HIGHER
PRIORITY THAN THIS! (higher priorities are lower numeric values. */
#define configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY 5

/* Interrupt priorities used by the kernel port layer itself.  These are generic
to all Cortex-M ports, and do not rely on any particular library functions. */
#define configKERNEL_INTERRUPT_PRIORITY (configLIBRARY_LOWEST_INTERRUPT_PRIORITY « (8 -
configPRIO_BITS))
/* !!!! configMAX_SYSCALL_INTERRUPT_PRIORITY must not be set to zero !!!!
See http://www.FreeRTOS.org/RTOS-Cortex-M3-M4.html. */
#define configMAX_SYSCALL_INTERRUPT_PRIORITY (configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY « (8 -
configPRIO_BITS))

/* Normal assert() semantics without relying on the provision of an assert.h
header file. */
#define configASSERT(x)            \
    if ((x) == 0)                  \
    {                              \
        taskDISABLE_INTERRUPTS();  \
        for (;;)                   \
            ;                      \
    }

/* Definitions that map the FreeRTOS port interrupt handlers to their CMSIS
standard names. */
#define vPortSVCHandler SVC_Handler
#define xPortPendSVHandler PendSV_Handler
#define xPortSysTickHandler SysTick_Handler

/* Functions and macros used for task statistics */
extern void bspConfigureTimForRunTimeStats(void);
extern uint16_t bspGetTimStatsCount(void);
#define portCONFIGURE_TIMER_FOR_RUN_TIME_STATS() bspConfigureTimForRunTimeStats()
#define portGET_RUN_TIME_COUNTER_VALUE() bspGetTimStatsCount();

#endif /* FREERTOS_CONFIG_H */
```

4.22 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↵RTOS/Core/Inc/main.h File Reference

main header file: Holds generic handlers.

Functions

- void Error_Handler (void)

  This function is executed in case of error occurrence.

4.22.1 Detailed Description

main header file: Holds generic handlers.

Author

Aaron Escoboza, Github account: https://github.com/aaron-ev

4.22.2 Function Documentation

4.22.2.1 Error_Handler()

```
void Error_Handler (
            void  )
```

This function is executed in case of error occurrence.

Return values

None	

4.23 main.h

Go to the documentation of this file.

```
#include "stm32f4xx_hal.h"

#ifndef __MAIN_H
#define __MAIN_H

void Error_Handler(void);

#endif
```

4.24 stm32f4xx_hal_conf.h

```
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_HAL_CONF_H
#define __STM32F4xx_HAL_CONF_H

#ifdef __cplusplus
 extern "C" {
#endif

/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/

/* ########################## Module Selection ############################## */
#define HAL_MODULE_ENABLED

  /* #define HAL_CRYP_MODULE_ENABLED */
/* #define HAL_ADC_MODULE_ENABLED */
/* #define HAL_CAN_MODULE_ENABLED */
/* #define HAL_CRC_MODULE_ENABLED */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED */
/* #define HAL_DAC_MODULE_ENABLED */
/* #define HAL_DCMI_MODULE_ENABLED */
/* #define HAL_DMA2D_MODULE_ENABLED */
/* #define HAL_ETH_MODULE_ENABLED */
/* #define HAL_ETH_LEGACY_MODULE_ENABLED */
/* #define HAL_NAND_MODULE_ENABLED */
/* #define HAL_NOR_MODULE_ENABLED */
/* #define HAL_PCCARD_MODULE_ENABLED */
/* #define HAL_SRAM_MODULE_ENABLED */
/* #define HAL_SDRAM_MODULE_ENABLED */
/* #define HAL_HASH_MODULE_ENABLED */
/* #define HAL_I2C_MODULE_ENABLED */
/* #define HAL_I2S_MODULE_ENABLED */
/* #define HAL_IWDG_MODULE_ENABLED */
/* #define HAL_LTDC_MODULE_ENABLED */
/* #define HAL_RNG_MODULE_ENABLED */
/* #define HAL_RTC_MODULE_ENABLED */
/* #define HAL_SAI_MODULE_ENABLED */
/* #define HAL_SD_MODULE_ENABLED */
/* #define HAL_MMC_MODULE_ENABLED */
/* #define HAL_SPI_MODULE_ENABLED */
#define HAL_TIM_MODULE_ENABLED
#define HAL_UART_MODULE_ENABLED
/* #define HAL_USART_MODULE_ENABLED */
/* #define HAL_IRDA_MODULE_ENABLED */
/* #define HAL_SMARTCARD_MODULE_ENABLED */
/* #define HAL_SMBUS_MODULE_ENABLED */
/* #define HAL_WWDG_MODULE_ENABLED */
/* #define HAL_PCD_MODULE_ENABLED */
/* #define HAL_HCD_MODULE_ENABLED */
/* #define HAL_DSI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_CEC_MODULE_ENABLED */
/* #define HAL_FMPI2C_MODULE_ENABLED */
/* #define HAL_FMPSMBUS_MODULE_ENABLED */
/* #define HAL_SPDIFRX_MODULE_ENABLED */
/* #define HAL_DFSDM_MODULE_ENABLED */
/* #define HAL_LPTIM_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_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_RTC_MODULE_ENABLED
/* ########################## HSE/HSI Values adaptation ##################### */
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    25000000U
#endif /* HSE_VALUE */

#if !defined  (HSE_STARTUP_TIMEOUT)
  #define HSE_STARTUP_TIMEOUT    100U
#endif /* HSE_STARTUP_TIMEOUT */

#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000U)
#endif /* HSI_VALUE */

#if !defined  (LSI_VALUE)
 #define LSI_VALUE  32000U
#endif /* LSI_VALUE */
#if !defined  (LSE_VALUE)
 #define LSE_VALUE  32768U
#endif /* LSE_VALUE */

#if !defined  (LSE_STARTUP_TIMEOUT)
  #define LSE_STARTUP_TIMEOUT    5000U
#endif /* LSE_STARTUP_TIMEOUT */

#if !defined  (EXTERNAL_CLOCK_VALUE)
  #define EXTERNAL_CLOCK_VALUE    12288000U
#endif /* EXTERNAL_CLOCK_VALUE */

/* Tip: To avoid modifying this file each time you need to use different HSE,
   ===  you can define the HSE value in your toolchain compiler preprocessor. */

/* ########################### System Configuration ######################### */
#define  VDD_VALUE            3300U
#define  TICK_INT_PRIORITY            15U
#define  USE_RTOS                     0U
#define  PREFETCH_ENABLE              1U
#define  INSTRUCTION_CACHE_ENABLE     1U
#define  DATA_CACHE_ENABLE            1U

#define  USE_HAL_ADC_REGISTER_CALLBACKS         0U /* ADC register callback disabled       */
#define  USE_HAL_CAN_REGISTER_CALLBACKS         0U /* CAN register callback disabled       */
#define  USE_HAL_CEC_REGISTER_CALLBACKS         0U /* CEC register callback disabled       */
#define  USE_HAL_CRYP_REGISTER_CALLBACKS        0U /* CRYP register callback disabled      */
#define  USE_HAL_DAC_REGISTER_CALLBACKS         0U /* DAC register callback disabled       */
```

```
#define  USE_HAL_DCMI_REGISTER_CALLBACKS        0U /* DCMI register callback disabled      */
#define  USE_HAL_DFSDM_REGISTER_CALLBACKS       0U /* DFSDM register callback disabled     */
#define  USE_HAL_DMA2D_REGISTER_CALLBACKS       0U /* DMA2D register callback disabled     */
#define  USE_HAL_DSI_REGISTER_CALLBACKS         0U /* DSI register callback disabled       */
#define  USE_HAL_ETH_REGISTER_CALLBACKS         0U /* ETH register callback disabled       */
#define  USE_HAL_HASH_REGISTER_CALLBACKS        0U /* HASH register callback disabled      */
#define  USE_HAL_HCD_REGISTER_CALLBACKS         0U /* HCD register callback disabled       */
#define  USE_HAL_I2C_REGISTER_CALLBACKS         0U /* I2C register callback disabled       */
#define  USE_HAL_FMPI2C_REGISTER_CALLBACKS      0U /* FMPI2C register callback disabled    */
#define  USE_HAL_FMPSMBUS_REGISTER_CALLBACKS    0U /* FMPSMBUS register callback disabled  */
#define  USE_HAL_I2S_REGISTER_CALLBACKS         0U /* I2S register callback disabled       */
#define  USE_HAL_IRDA_REGISTER_CALLBACKS        0U /* IRDA register callback disabled      */
#define  USE_HAL_LPTIM_REGISTER_CALLBACKS       0U /* LPTIM register callback disabled     */
#define  USE_HAL_LTDC_REGISTER_CALLBACKS        0U /* LTDC register callback disabled      */
#define  USE_HAL_MMC_REGISTER_CALLBACKS         0U /* MMC register callback disabled       */
#define  USE_HAL_NAND_REGISTER_CALLBACKS        0U /* NAND register callback disabled      */
#define  USE_HAL_NOR_REGISTER_CALLBACKS         0U /* NOR register callback disabled       */
#define  USE_HAL_PCCARD_REGISTER_CALLBACKS      0U /* PCCARD register callback disabled    */
#define  USE_HAL_PCD_REGISTER_CALLBACKS         0U /* PCD register callback disabled       */
#define  USE_HAL_QSPI_REGISTER_CALLBACKS        0U /* QSPI register callback disabled      */
#define  USE_HAL_RNG_REGISTER_CALLBACKS         0U /* RNG register callback disabled       */
#define  USE_HAL_RTC_REGISTER_CALLBACKS         0U /* RTC register callback disabled       */
#define  USE_HAL_SAI_REGISTER_CALLBACKS         0U /* SAI register callback disabled       */
#define  USE_HAL_SD_REGISTER_CALLBACKS          0U /* SD register callback disabled        */
#define  USE_HAL_SMARTCARD_REGISTER_CALLBACKS   0U /* SMARTCARD register callback disabled */
#define  USE_HAL_SDRAM_REGISTER_CALLBACKS       0U /* SDRAM register callback disabled     */
#define  USE_HAL_SRAM_REGISTER_CALLBACKS        0U /* SRAM register callback disabled      */
#define  USE_HAL_SPDIFRX_REGISTER_CALLBACKS     0U /* SPDIFRX register callback disabled   */
#define  USE_HAL_SMBUS_REGISTER_CALLBACKS       0U /* SMBUS register callback disabled     */
#define  USE_HAL_SPI_REGISTER_CALLBACKS         0U /* SPI register callback disabled       */
#define  USE_HAL_TIM_REGISTER_CALLBACKS         0U /* TIM register callback disabled       */
#define  USE_HAL_UART_REGISTER_CALLBACKS        0U /* UART register callback disabled      */
#define  USE_HAL_USART_REGISTER_CALLBACKS       0U /* USART register callback disabled     */
#define  USE_HAL_WWDG_REGISTER_CALLBACKS        0U /* WWDG register callback disabled      */

/* ########################## Assert Selection ############################## */
/* #define USE_FULL_ASSERT    1U */

/* ################## Ethernet peripheral configuration ##################### */

/* Section 1 : Ethernet peripheral configuration */

/* MAC ADDRESS: MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */
#define MAC_ADDR0   2U
#define MAC_ADDR1   0U
#define MAC_ADDR2   0U
#define MAC_ADDR3   0U
#define MAC_ADDR4   0U
#define MAC_ADDR5   0U

/* Definition of the Ethernet driver buffers size and count */
#define ETH_RX_BUF_SIZE                /* buffer size for receive               */
#define ETH_TX_BUF_SIZE                ETH_MAX_PACKET_SIZE /* buffer size for transmit              */
#define ETH_RXBUFNB                    4U       /* 4 Rx buffers of size ETH_RX_BUF_SIZE  */
#define ETH_TXBUFNB                    4U       /* 4 Tx buffers of size ETH_TX_BUF_SIZE  */

/* Section 2: PHY configuration section */

/* DP83848_PHY_ADDRESS Address*/
#define DP83848_PHY_ADDRESS           0x01U
/* PHY Reset delay these values are based on a 1 ms Systick interrupt*/
#define PHY_RESET_DELAY                 0x000000FFU
/* PHY Configuration delay */
#define PHY_CONFIG_DELAY                0x00000FFFU

#define PHY_READ_TO                     0x0000FFFFU
#define PHY_WRITE_TO                    0x0000FFFFU

/* Section 3: Common PHY Registers */

#define PHY_BCR                         ((uint16_t)0x0000U)
#define PHY_BSR                         ((uint16_t)0x0001U)
#define PHY_RESET                       ((uint16_t)0x8000U)
#define PHY_LOOPBACK                    ((uint16_t)0x4000U)
#define PHY_FULLDUPLEX_100M             ((uint16_t)0x2100U)
#define PHY_HALFDUPLEX_100M             ((uint16_t)0x2000U)
#define PHY_FULLDUPLEX_10M              ((uint16_t)0x0100U)
#define PHY_HALFDUPLEX_10M              ((uint16_t)0x0000U)
#define PHY_AUTONEGOTIATION             ((uint16_t)0x1000U)
#define PHY_RESTART_AUTONEGOTIATION     ((uint16_t)0x0200U)
#define PHY_POWERDOWN                   ((uint16_t)0x0800U)
#define PHY_ISOLATE                     ((uint16_t)0x0400U)
#define PHY_AUTONEGO_COMPLETE           ((uint16_t)0x0020U)
#define PHY_LINKED_STATUS               ((uint16_t)0x0004U)
#define PHY_JABBER_DETECTION            ((uint16_t)0x0002U)
/* Section 4: Extended PHY Registers */
#define PHY_SR                          ((uint16_t)0x10U)
```

```
#define PHY_SPEED_STATUS                ((uint16_t)0x0002U)
#define PHY_DUPLEX_STATUS               ((uint16_t)0x0004U)
/* ################## SPI peripheral configuration ########################## */

/* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver
* Activated: CRC code is present inside driver
* Deactivated: CRC code cleaned from driver
*/

#define USE_SPI_CRC                     0U

/* Includes ------------------------------------------------------------------*/
#ifdef HAL_RCC_MODULE_ENABLED
  #include "stm32f4xx_hal_rcc.h"
#endif /* HAL_RCC_MODULE_ENABLED */

#ifdef HAL_GPIO_MODULE_ENABLED
  #include "stm32f4xx_hal_gpio.h"
#endif /* HAL_GPIO_MODULE_ENABLED */

#ifdef HAL_EXTI_MODULE_ENABLED
  #include "stm32f4xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */

#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32f4xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */

#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32f4xx_hal_cortex.h"
#endif /* HAL_CORTEX_MODULE_ENABLED */

#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f4xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */

#ifdef HAL_CAN_MODULE_ENABLED
  #include "stm32f4xx_hal_can.h"
#endif /* HAL_CAN_MODULE_ENABLED */

#ifdef HAL_CAN_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_can_legacy.h"
#endif /* HAL_CAN_LEGACY_MODULE_ENABLED */

#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32f4xx_hal_crc.h"
#endif /* HAL_CRC_MODULE_ENABLED */

#ifdef HAL_CRYP_MODULE_ENABLED
  #include "stm32f4xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */

#ifdef HAL_DMA2D_MODULE_ENABLED
  #include "stm32f4xx_hal_dma2d.h"
#endif /* HAL_DMA2D_MODULE_ENABLED */

#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32f4xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */

#ifdef HAL_DCMI_MODULE_ENABLED
  #include "stm32f4xx_hal_dcmi.h"
#endif /* HAL_DCMI_MODULE_ENABLED */

#ifdef HAL_ETH_MODULE_ENABLED
  #include "stm32f4xx_hal_eth.h"
#endif /* HAL_ETH_MODULE_ENABLED */

#ifdef HAL_ETH_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_eth_legacy.h"
#endif /* HAL_ETH_LEGACY_MODULE_ENABLED */

#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32f4xx_hal_flash.h"
#endif /* HAL_FLASH_MODULE_ENABLED */

#ifdef HAL_SRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sram.h"
#endif /* HAL_SRAM_MODULE_ENABLED */

#ifdef HAL_NOR_MODULE_ENABLED
  #include "stm32f4xx_hal_nor.h"
#endif /* HAL_NOR_MODULE_ENABLED */

#ifdef HAL_NAND_MODULE_ENABLED
  #include "stm32f4xx_hal_nand.h"
#endif /* HAL_NAND_MODULE_ENABLED */
```

```

#ifdef HAL_PCCARD_MODULE_ENABLED
  #include "stm32f4xx_hal_pccard.h"
#endif /* HAL_PCCARD_MODULE_ENABLED */

#ifdef HAL_SDRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sdram.h"
#endif /* HAL_SDRAM_MODULE_ENABLED */

#ifdef HAL_HASH_MODULE_ENABLED
 #include "stm32f4xx_hal_hash.h"
#endif /* HAL_HASH_MODULE_ENABLED */

#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32f4xx_hal_i2c.h"
#endif /* HAL_I2C_MODULE_ENABLED */

#ifdef HAL_SMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */

#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32f4xx_hal_i2s.h"
#endif /* HAL_I2S_MODULE_ENABLED */

#ifdef HAL_IWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */

#ifdef HAL_LTDC_MODULE_ENABLED
 #include "stm32f4xx_hal_ltdc.h"
#endif /* HAL_LTDC_MODULE_ENABLED */

#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32f4xx_hal_pwr.h"
#endif /* HAL_PWR_MODULE_ENABLED */

#ifdef HAL_RNG_MODULE_ENABLED
 #include "stm32f4xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */

#ifdef HAL_RTC_MODULE_ENABLED
 #include "stm32f4xx_hal_rtc.h"
#endif /* HAL_RTC_MODULE_ENABLED */

#ifdef HAL_SAI_MODULE_ENABLED
 #include "stm32f4xx_hal_sai.h"
#endif /* HAL_SAI_MODULE_ENABLED */

#ifdef HAL_SD_MODULE_ENABLED
 #include "stm32f4xx_hal_sd.h"
#endif /* HAL_SD_MODULE_ENABLED */

#ifdef HAL_SPI_MODULE_ENABLED
 #include "stm32f4xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */

#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32f4xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */

#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32f4xx_hal_uart.h"
#endif /* HAL_UART_MODULE_ENABLED */

#ifdef HAL_USART_MODULE_ENABLED
 #include "stm32f4xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */

#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32f4xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */

#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32f4xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */

#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_wwdg.h"
#endif /* HAL_WWDG_MODULE_ENABLED */

#ifdef HAL_PCD_MODULE_ENABLED
 #include "stm32f4xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */

#ifdef HAL_HCD_MODULE_ENABLED
 #include "stm32f4xx_hal_hcd.h"
```

```
#endif /* HAL_HCD_MODULE_ENABLED */

#ifdef HAL_DSI_MODULE_ENABLED
 #include "stm32f4xx_hal_dsi.h"
#endif /* HAL_DSI_MODULE_ENABLED */

#ifdef HAL_QSPI_MODULE_ENABLED
 #include "stm32f4xx_hal_qspi.h"
#endif /* HAL_QSPI_MODULE_ENABLED */

#ifdef HAL_CEC_MODULE_ENABLED
 #include "stm32f4xx_hal_cec.h"
#endif /* HAL_CEC_MODULE_ENABLED */

#ifdef HAL_FMPI2C_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpi2c.h"
#endif /* HAL_FMPI2C_MODULE_ENABLED */

#ifdef HAL_FMPSMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpsmbus.h"
#endif /* HAL_FMPSMBUS_MODULE_ENABLED */

#ifdef HAL_SPDIFRX_MODULE_ENABLED
 #include "stm32f4xx_hal_spdifrx.h"
#endif /* HAL_SPDIFRX_MODULE_ENABLED */

#ifdef HAL_DFSDM_MODULE_ENABLED
 #include "stm32f4xx_hal_dfsdm.h"
#endif /* HAL_DFSDM_MODULE_ENABLED */

#ifdef HAL_LPTIM_MODULE_ENABLED
 #include "stm32f4xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */

#ifdef HAL_MMC_MODULE_ENABLED
 #include "stm32f4xx_hal_mmc.h"
#endif /* HAL_MMC_MODULE_ENABLED */

/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
/* Exported functions ------------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
#else
  #define assert_param(expr) ((void)0U)
#endif /* USE_FULL_ASSERT */

#ifdef __cplusplus
}
#endif

#endif /* __STM32F4xx_HAL_CONF_H */
```

4.25 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩RTOS/Core/Inc/stm32f4xx_it.h File Reference

Interrupt header file: Holds interrupt handlers.

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 **DebugMon_Handler** (void)

  This function handles Debug monitor.
- void **TIM1_BRK_TIM9_IRQHandler** (void)

  This function handles TIM1 break interrupt and TIM9 global interrupt.

4.25.1 Detailed Description

Interrupt header file: Holds interrupt handlers.

Author

Aaron Escoboza, Github account: `https://github.com/aaron-ev`

4.26 stm32f4xx_it.h

Go to the documentation of this file.
```
#ifndef __STM32F4xx_IT_H
#define __STM32F4xx_IT_H

void NMI_Handler(void);
void HardFault_Handler(void);
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void DebugMon_Handler(void);
void TIM1_BRK_TIM9_IRQHandler(void);

#endif
```

4.27 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩ RTOS/Core/Src/console.c File Reference

Command Line Interpreter based on FreeRTOS and STM32 HAL layer.

Macros

- #define **CONSOLE_VERSION_MAJOR** 1
- #define **CONSOLE_VERSION_MINOR** 0
- #define **MAX_IN_STR_LEN** 300
- #define **MAX_OUT_STR_LEN** 600
- #define **MAX_RX_QUEUE_LEN** 300
- #define **ASCII_TAB** '\t' /∗ Tabulate ∗/
- #define **ASCII_CR** '\r' /∗ Carriage return ∗/
- #define **ASCII_LF** '\n' /∗ Line feed ∗/
- #define **ASCII_BACKSPACE** '\b' /∗ Back space ∗/
- #define **ASCII_FORM_FEED** '\f' /∗ Form feed ∗/
- #define **ASCII_DEL** 127 /∗ Delete ∗/
- #define **ASCII_CTRL_PLUS_C** 3 /∗ CTRL + C ∗/
- #define **ASCII_NACK** 21 /∗ Negative acknowledge ∗/

Functions

- void vConsoleEnableRxInterrupt (void)

    Enables UART RX reception.
- void vTaskConsole (void ∗pvParams)

    Task to handle user commands via serial communication.
- BaseType_t xbspConsoleInit (uint16_t usStackSize, UBaseType_t uxPriority, UART_HandleTypeDef ∗px←
UartHandle)

    Initialize the console by registering all commands and creating a task.
- void HAL_UART_RxCpltCallback (UART_HandleTypeDef ∗huart)

    Callback for UART RX, triggered any time there is a new character.

Variables

- char **cRxData**
- QueueHandle_t **xQueueRxHandle**
- UART_HandleTypeDef ∗ **pxUartDevHandle**

4.27.1 Detailed Description

Command Line Interpreter based on FreeRTOS and STM32 HAL layer.

Author

Aaron Escoboza, Github account: `https://github.com/aaron-ev`

4.27.2 Function Documentation

4.27.2.1 HAL_UART_RxCpltCallback()

```
void HAL_UART_RxCpltCallback (
            UART_HandleTypeDef ∗ huart )
```

Callback for UART RX, triggered any time there is a new character.

Parameters

∗*huart*	Pointer to the uart handle.

Return values

void	

4.27.2.2 vConsoleEnableRxInterrupt()

```
void vConsoleEnableRxInterrupt (
            void )
```

Enables UART RX reception.

Parameters

void	

Return values

void	

4.27.2.3 vTaskConsole()

```
void vTaskConsole (
            void * pvParams )
```

Task to handle user commands via serial communication.

Parameters

∗pvParams	Data passed at task creation.

Return values

void	

4.27.2.4 xbspConsoleInit()

```
BaseType_t xbspConsoleInit (
            uint16_t usStackSize,
            UBaseType_t uxPriority,
            UART_HandleTypeDef * pxUartHandle )
```

Initialize the console by registering all commands and creating a task.

Parameters

usStackSize	Task console stack size
uxPriority	Task console priority
∗pxUartHandle	Pointer for uart handle.

Return values

FreeRTOS	status

4.28 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩ RTOS/Core/Src/main.c File Reference

Command Line Interpreter based on FreeRTOS and STM32 HAL layer.

Functions

- void vTaskHeartBeat (void ∗pvParams)

  Heart beat task indicates project alive by toggling an LED.
- int main (void)

  main function: Initialize BSP, console, and FreeRTOS.
- 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

- TaskHandle_t **xTaskHeartBeatHandler**
- UART_HandleTypeDef **consoleHandle**

4.28.1 Detailed Description

Command Line Interpreter based on FreeRTOS and STM32 HAL layer.

Author

Aaron Escoboza, Github account: `https://github.com/aaron-ev`

4.28.2 Function Documentation

4.28.2.1 Error_Handler()

```
void Error_Handler (
             void   )
```

This function is executed in case of error occurrence.

Return values

None	

4.28.2.2 HAL_TIM_PeriodElapsedCallback()

```
void HAL_TIM_PeriodElapsedCallback (
             TIM_HandleTypeDef ∗ htim )
```

Period elapsed callback in non blocking mode.

Note

This function is called when TIM9 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	

4.28.2.3 main()

```
int main (
            void  )
```

main function: Initialize BSP, console, and FreeRTOS.

Parameters

void	

Return values

void	

4.28.2.4 vTaskHeartBeat()

```
void vTaskHeartBeat (
            void * pvParams )
```

Heart beat task indicates project alive by toggling an LED.

Parameters

∗pvParams	data passed at task creation

Return values

void	

4.29 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩ RTOS/Core/Src/msp.c File Reference

Interrupt header file: Holds interrupt handlers.

Functions

- void HAL_MspInit (void)

  Enable peripheral clocks and set NVIC priorities.
- void HAL_UART_MspInit (UART_HandleTypeDef ∗uartHandler)

  Low level initialization for console UART.
- void HAL_TIM_OC_MspInit (TIM_HandleTypeDef ∗timerHandler)

  Low level initialization for GPIO pins assigned to PWM feature.
- void HAL_RTC_MspInit (RTC_HandleTypeDef ∗rtcHandler)

  Low level initialization for RTC.

4.29.1 Detailed Description

Interrupt header file: Holds interrupt handlers.

Author

Aaron Escoboza, Github account: `https://github.com/aaron-ev`

4.29.2 Function Documentation

4.29.2.1 HAL_MspInit()

```
void HAL_MspInit (
            void )
```

Enable peripheral clocks and set NVIC priorities.

Parameters

void	

Return values

void	

4.29.2.2 HAL_RTC_MspInit()

```
void HAL_RTC_MspInit (
            RTC_HandleTypeDef ∗ rtcHandler )
```

Low level initialization for RTC.

Parameters

∗*rtcHandler*	RTC handler

Return values

void	

4.29.2.3 HAL_TIM_OC_MspInit()

```
void HAL_TIM_OC_MspInit (
            TIM_HandleTypeDef * timerHandler )
```

Low level initialization for GPIO pins assigned to PWM feature.

Parameters

∗*timerHandler*	Timer handler assigned to PWM feature.

Return values

void	

4.29.2.4 HAL_UART_MspInit()

```
void HAL_UART_MspInit (
            UART_HandleTypeDef * uartHandler )
```

Low level initialization for console UART.

Parameters

∗*uartHandler*	UART handler that would be used for the console

Return values

void	

4.30 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩ RTOS/Core/Src/stm32f4xx_hal_timebase_tim.c File Reference

HAL time base based on the hardware TIM.

Functions

- HAL_StatusTypeDef HAL_InitTick (uint32_t TickPriority)

  This function configures the TIM9 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 **htim9**

4.30.1 Detailed Description

HAL time base based on the hardware TIM.

Attention

4.30.2 Function Documentation

4.30.2.1 HAL_InitTick()

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

This function configures the TIM9 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

4.30.2.2 HAL_ResumeTick()

```
void HAL_ResumeTick (
             void  )
```

Resume Tick increment.

Note

Enable the tick increment by Enabling TIM9 update interrupt.

Parameters

None	

Return values

None	

4.30.2.3 HAL_SuspendTick()

```
void HAL_SuspendTick (
             void  )
```

Suspend Tick increment.

Note

Disable the tick increment by disabling TIM9 update interrupt.

Parameters

None	

Return values

None	

4.31 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩ RTOS/Core/Src/stm32f4xx_it.c File Reference

Interrupt source file: Holds interrupt handler implementations.

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 **DebugMon_Handler** (void)

  This function handles Debug monitor.
- void **TIM1_BRK_TIM9_IRQHandler** (void)

  This function handles TIM1 break interrupt and TIM9 global interrupt.
- void **USART1_IRQHandler** (void)

  This function handles UART1 interrupts.
- void **TIM2_IRQHandler** (void)

  This function handles TIM2 interrupts.

Variables

- TIM_HandleTypeDef **htim9**
- UART_HandleTypeDef **consoleHandle**

4.31.1 Detailed Description

Interrupt source file: Holds interrupt handler implementations.

Author

Aaron Escoboza, Github account: `https://github.com/aaron-ev`

4.32 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩ RTOS/Core/Src/syscalls.c File Reference

STM32CubeIDE Minimal System calls file.

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

4.32.1 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

4.33 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩ RTOS/Core/Src/sysmem.c File Reference

STM32CubeIDE System Memory calls file.

Functions

- void ∗ _sbrk (ptrdiff_t incr)

  _sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

4.33.1 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

4.33.2 Function Documentation

4.33.2.1 _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

4.34 C:/Users/aaron/main/projects/cli_freeRTOS/workspace/cliFree↩ RTOS/Core/Src/system_stm32f4xx.c File Reference

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

Macros

- #define HSE_VALUE ((uint32_t)25000000)
- #define HSI_VALUE ((uint32_t)16000000)

Functions

- void SystemInit (void)

  Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.
- 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** = 16000000
- 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}

4.34.1 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_stm32f4xx.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.

Attention

Index