FreeRTOS CLI + STM32 -- By Aaron Escoboza

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(∗)** (p. 6)

> - If SYSCLK source is HSE, SystemCoreClock will contain the **HSE_VALUE(∗∗)** (p. 6)

> - If SYSCLK source is PLL, SystemCoreClock will contain the **HSE_VALUE(∗∗)** (p. 6) or **HSI_VALUE(∗)** (p. 6) multiplied/divided by the PLL factors.

(∗) HSI_VALUE is a constant defined in **stm32f4xx_hal_conf.h** (p. 31) 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** (p. 31) 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

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.
```
00001
00009 #ifndef __BSP__H
00010 #define __BSP__H
00011
00012 #include "bspPwm.h"
00013 #include "bspGpio.h"
00014 #include "bspClk.h"
00015
00016 BspError_e bspInit(void);
00017
00018 #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 **bspGetClockInfo** (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

4.3.2 Function Documentation

4.3.2.1 bspGetClockInfo()

```
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.
```
00001
00009 #ifndef __BSP_CLK_H
00010 #define __BSP_CLK_H
00011
00012
00013 #include "stdio.h"
00014 #include "stdint.h"
00015 #include "stm32f4xx_hal.h"
00016
00017 void bspGetClockIinfo(char *pcWriteBuffer, size_t xWriteBufferLen);
00018
00019 #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

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.
```
00001
00009 #ifndef __BSP_GPIO_H
00010 #define __BSP_GPIO_H
00011
00012 #include "stdint.h"
00013 #include "stm32f4xx_hal.h"
00014
00015 typedef enum
00016 {
00017     BSP_GPIOA,
00018     BSP_GPIOB,
00019     BSP_GPIOC,
00020     BSP_GPIOD,
00021     BSP_GPIOE,
00022     BSP_GPIOH,
00023     BSP_MAX_GPIO_INSTANCE,
00024 }BspGpioInstance_e;
00025
00026 typedef enum
00027 {
00028     BSP_GPIO_PIN_0,
00029     BSP_GPIO_PIN_1,
00030     BSP_GPIO_PIN_2,
```

```
00031    BSP_GPIO_PIN_3,
00032    BSP_GPIO_PIN_4,
00033    BSP_GPIO_PIN_5,
00034    BSP_GPIO_PIN_6,
00035    BSP_GPIO_PIN_7,
00036    BSP_GPIO_PIN_8,
00037    BSP_GPIO_PIN_9,
00038    BSP_GPIO_PIN_10,
00039    BSP_GPIO_PIN_11,
00040    BSP_GPIO_PIN_12,
00041    BSP_GPIO_PIN_13,
00042    BSP_GPIO_PIN_14,
00043    BSP_GPIO_PIN_15,
00044 }BspPinNum_e;
00045
00046 typedef enum
00047 {
00048    BSP_GPIO_PIN_LOW,
00049    BSP_GPIO_PIN_HIGH,
00050 }BspGpioPinState_e;
00051
00052 void bspGpioToggle(BspGpioInstance_e eGpio, BspPinNum_e pinNum);
00053 BspGpioInstance_e bspGpioMapInstance(const char pcGpioInstance);
00054 BspGpioPinState_e bspGpioRead(BspGpioInstance_e eGpio, BspPinNum_e pinNum);
00055 void bspGpioWrite(BspGpioInstance_e eGpio, BspPinNum_e pinNum, BspGpioPinState_e pinState);
00056
00057 #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

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.
```
00001
00009 #ifndef __BSP_PWM_H
00010 #define __BSP_PWM_H
00011
00012 #include "stdint.h"
00013 #include "stm32f4xx_hal.h"
00014 #include "bspTypeDef.h"
00015
00016 typedef enum
00017 {
00018     PWM_CH_1,
00019     PWM_CH_2,
00020     PWM_CH_3,
00021     PWM_CH_4,
00022     MAX_PWM_CH,
00023 } pwmChannels_e;
```

```
00024
00025 BspError_e bspPwmInit(void);
00026 TIM_HandleTypeDef* bspPwmGetHandler(void);
00027 BspError_e bspPwmSetFreq(uint32_t uNewFreq);
00028 void bspPwmStart(pwmChannels_e eChannelIndex);
00029 BspError_e bspPwmSetDuty(uint8_t uNewDuty, pwmChannels_e xChannel);
00030
00031 #endif
```

4.9 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 }

4.9.1 Detailed Description

Header that contains BSP definitions.

Author

Aaron Escoboza

4.10 bspTypeDef.h

Go to the documentation of this file.
```
00001
00009 #ifndef __BSP_TYPE_DEF_H
00010 #define __BSP_TYPE_DEF_H
00011
00012 #include "errno.h"
00013
00014 typedef enum
00015
00016 {
00017     BSP_NO_ERROR,
00018     BSP_ERROR_EIO = EIO,
00019     BSP_ERROR_EINVAL = EINVAL,
00020 } BspError_e;
00021
00022 #endif
```

4.11 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

- static BspError_e **clkInit** (void)

  Initialize system clocks, PLL and Clock dividers.
- static void **heartBeatInit** (void)

  Initialize GPIO pin for heart beat functionality.
- void **bspConfigureTimForRunTimeStats** (void)

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

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

  Initialize UART frame.
- BspError_e **bspInit** (void)

  Calls all BSP init functions.

Variables

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

4.11.1   Detailed Description

source file to implement low level initializations.

Author

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

4.11.2   Function Documentation

4.11.2.1   bspConfigureTimForRunTimeStats()

```
void bspConfigureTimForRunTimeStats (
           void  )
```

Configure timer used for FreeRTOS task statistics.

Parameters

void	

Return values

void	

4.11.2.2 bspGetTimStatsCount()

```
uint32_t bspGetTimStatsCount (
            void  )
```

Get current timer counter for FreeRTOS task statistics.

Parameters

void	

Return values

void	

4.11.2.3 bspInit()

```
BspError_e bspInit (
            void  )
```

Calls all BSP init functions.

Parameters

void	

Return values

BspError↩ _e	

4.11.2.4 clkInit()

```
static BspError_e clkInit (
            void  )  [static]
```

Initialize system clocks, PLL and Clock dividers.

Parameters

void	

Return values

BspError↩ _e	

4.11.2.5 consoleInit()

```
BspError_e consoleInit (
            void )
```

Initialize UART frame.

Parameters

void	

Return values

BSP	error

4.11.2.6 heartBeatInit()

```
static void heartBeatInit (
            void ) [static]
```

Initialize GPIO pin for heart beat functionality.

Parameters

void	

Return values

void	

4.12 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 **bspGetClockInfo** (char ∗pcWriteBuffer, size_t xWriteBufferLen)

  Gets system clock, PCLKx and CLK dividers.

4.12.1 Detailed Description

source file to implement functions related to the clock tree.

Author

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

4.12.2 Function Documentation

4.12.2.1 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	

4.13 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

- static uint16_t **bspMapPinNumFromBspToHal** (uint16_t uGpioNumber)

    Maps a number from the range 0 - 15 to STM32 pin macro value.
- BspGpioInstance_e **bspGpioMapInstance** (const char pcGpioInstance)

    Maps a letter (a, b, c, d, e , h) to a BSP GPIO instance.
- static GPIO_TypeDef ∗ **bspMapInstanceToHal** (BspGpioInstance_e eGpio)

    Maps from a BSP instance to HAL GPIO instance pointer.
- 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.

4.13.1 Detailed Description

source file to implement functions related to GPIO operations.

Author

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

4.13.2 Function Documentation

4.13.2.1 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.

4.13.2.2 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

4.13.2.3 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	

4.13.2.4 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	

4.13.2.5 bspMapInstanceToHal()

```
static GPIO_TypeDef * bspMapInstanceToHal (
            BspGpioInstance_e eGpio )  [static]
```

Maps from a BSP instance to HAL GPIO instance pointer.

Parameters

eGpio	BSP GPIO instance.

Return values

HAL	GPIO instance pointer.

4.13.2.6 bspMapPinNumFromBspToHal()

```
static uint16_t bspMapPinNumFromBspToHal (
            uint16_t uGpioNumber )  [static]
```

Maps a number from the range 0 - 15 to STM32 pin macro value.

Parameters

uGpioNumber	Pin number.

Return values

STM32	HAL pin number.

4.14 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
- #define **PWM_TIM_CHANNEL** TIM_CHANNEL_1
- #define **PWM_DEFAULT_PULSE** 500

4.14.1 Detailed Description

Hold general application configuration.

Author

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

4.15 appConfig.h

Go to the documentation of this file.
```
00001
00009 #ifndef APP_CONFIG__H
00010 #define APP_CONFIG__H
00011
00012 /* Enable debug messages */
00013 #define DEBUG_PRINT_EN                  1 /* 1 = Enable , 0 =  Disable */
00014
00015 /* Task priorities */
00016 #define HEART_BEAT_PRIORITY_TASK        1
00017
00018 /* Heart beat settings */
00019 #define HEART_BEAT_LED_PORT             GPIOC
00020 #define HEART_BEAT_LED_PIN              GPIO_PIN_13
00021 #define HEART_BEAT_BLINK_DELAY          500 /* In ms */
00022
00023 /* CLI console settings */
00024 #define CONSOLE_INSTANCE                USART1
00025 #define CONSOLE_TX_PIN                  GPIO_PIN_6
00026 #define CONSOLE_RX_PIN                  GPIO_PIN_7
00027 #define CONSOLE_GPIO_PORT               GPIOB
00028 #define CONSOLE_BAUDRATE                9600
00029 #define CONSOLE_TASK_PRIORITY           1
00030 #define CONSOLE_STACK_SIZE              3000
00031
00032 /* PWM signal settings */
00033 #define PWM_GPIO_INSTANCE               GPIOA
00034 #define PWM_GPIO_PINX                   GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3
00035 #define PWM_GPIO_ALTERNATE              GPIO_AF1_TIM2
00036 #define PWM_TIM_INSTANCE                TIM2
00037 #define PWM_TIM_CHANNEL                 TIM_CHANNEL_1
00038 #define PWM_DEFAULT_PULSE               500
00039
00040 #endif
```

4.16 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 **xConsoleInit** (uint16_t usStackSize, UBaseType_t uxPriority, UART_HandleTypeDef ∗pxUart↩ Handle)

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

4.16.1 Detailed Description

Console header file: APIs to handle the console.

Author

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

4.16.2 Function Documentation

4.16.2.1 xConsoleInit()

```
BaseType_t xConsoleInit (
          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

4.17 console.h

Go to the documentation of this file.
```
00001
00009 #ifndef __CONSOLE__H
00010 #define __CONSOLE__H
00011
00012 #include "FreeRTOS.h"
00013
00014 BaseType_t xConsoleInit(uint16_t usStackSize, UBaseType_t uxPriority, UART_HandleTypeDef
      *pxUartHandle);
00015
00016 #endif
```

4.18 FreeRTOSConfig.h

```
00001 /*
00002  * FreeRTOS V202112.00
00003  * Copyright (C) 2020 Amazon.com, Inc. or its affiliates.  All Rights Reserved.
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy of
00006  * this software and associated documentation files (the "Software"), to deal in
00007  * the Software without restriction, including without limitation the rights to
00008  * use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
00009  * the Software, and to permit persons to whom the Software is furnished to do so,
00010  * subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
00017  * FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
00018  * COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
00019  * IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
00020  * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00021  *
00022  * http://www.FreeRTOS.org
00023  * http://aws.amazon.com/freertos
00024  *
00025  * 1 tab == 4 spaces!
00026  */
00027
00028 #ifndef FREERTOS_CONFIG_H
00029 #define FREERTOS_CONFIG_H
00030
00031 /*-----------------------------------------------------------
00032  * Application specific definitions.
00033  *
00034  * These definitions should be adjusted for your particular hardware and
00035  * application requirements.
00036  *
00037  * THESE PARAMETERS ARE DESCRIBED WITHIN THE 'CONFIGURATION' SECTION OF THE
00038  * FreeRTOS API DOCUMENTATION AVAILABLE ON THE FreeRTOS.org WEB SITE.
00039  *
00040  * See http://www.freertos.org/a00110.html
00041  *----------------------------------------------------------*/
00042
00043 /* Ensure stdint is only used by the compiler, and not the assembler. */
```

```
00044 #if defined(__ICCARM__) || defined(__GNUC__) || defined(_CC_ARM)
00045     #include <stdint.h>
00046     extern uint32_t SystemCoreClock;
00047 #endif
00048
00049 #define configUSE_PREEMPTION 1
00050 #define configUSE_IDLE_HOOK 0
00051 #define configUSE_TICK_HOOK 0
00052 #define configCPU_CLOCK_HZ (SystemCoreClock)
00053 #define configTICK_RATE_HZ ((TickType_t)1000)
00054 #define configMAX_PRIORITIES (5)
00055 #define configMINIMAL_STACK_SIZE ((unsigned short)130)
00056 //#define configTOTAL_HEAP_SIZE ((size_t)(75 * 1024))
00057 #define configTOTAL_HEAP_SIZE ((size_t)(75 * 524))
00058 #define configMAX_TASK_NAME_LEN (10)
00059 #define configUSE_TRACE_FACILITY 1
00060 #define configUSE_16_BIT_TICKS 0
00061 #define configIDLE_SHOULD_YIELD 1
00062 #define configUSE_MUTEXES 1
00063 #define configQUEUE_REGISTRY_SIZE 8
00064 #define configCHECK_FOR_STACK_OVERFLOW 0
00065 #define configUSE_RECURSIVE_MUTEXES 1
00066 #define configUSE_MALLOC_FAILED_HOOK 0
00067 #define configUSE_APPLICATION_TASK_TAG 0
00068 #define configUSE_COUNTING_SEMAPHORES 1
00069 #define configGENERATE_RUN_TIME_STATS 1
00070 #define configUSE_TASK_NOTIFICATIONS 1
00071 #define configTASK_NOTIFICATION_ARRAY_ENTRIES 2
00072 #define configCOMMAND_INT_MAX_OUTPUT_SIZE 500
00073
00074 /* Co-routine definitions. */
00075 #define configUSE_CO_ROUTINES 0
00076 #define configMAX_CO_ROUTINE_PRIORITIES (2)
00077
00078 /* Software timer definitions. */
00079 #define configUSE_TIMERS 1
00080 #define configTIMER_TASK_PRIORITY (2)
00081 #define configTIMER_QUEUE_LENGTH 10
00082 #define configTIMER_TASK_STACK_DEPTH (configMINIMAL_STACK_SIZE * 2)
00083
00084 /* Set the following definitions to 1 to include the API function, or zero
00085 to exclude the API function. */
00086 #define INCLUDE_vTaskPrioritySet 1
00087 #define INCLUDE_uxTaskPriorityGet 1
00088 #define INCLUDE_vTaskDelete 1
00089 #define INCLUDE_vTaskCleanUpResources 1
00090 #define INCLUDE_vTaskSuspend 1
00091 #define INCLUDE_vTaskDelayUntil 1
00092 #define INCLUDE_vTaskDelay 1
00093
00094 /* Cortex-M specific definitions. */
00095 #ifdef __NVIC_PRIO_BITS
00096 /* __BVIC_PRIO_BITS will be specified when CMSIS is being used. */
00097 #define configPRIO_BITS __NVIC_PRIO_BITS
00098 #else
00099 #define configPRIO_BITS 4 /* 15 priority levels */
00100 #endif
00101
00102 /* The lowest interrupt priority that can be used in a call to a "set priority"
00103 function. */
00104 #define configLIBRARY_LOWEST_INTERRUPT_PRIORITY 0xf
00105
00106 /* The highest interrupt priority that can be used by any interrupt service
00107 routine that makes calls to interrupt safe FreeRTOS API functions. DO NOT CALL
00108 INTERRUPT SAFE FREERTOS API FUNCTIONS FROM ANY INTERRUPT THAT HAS A HIGHER
00109 PRIORITY THAN THIS! (higher priorities are lower numeric values. */
00110 #define configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY 5
00111
00112 /* Interrupt priorities used by the kernel port layer itself. These are generic
00113 to all Cortex-M ports, and do not rely on any particular library functions. */
00114 #define configKERNEL_INTERRUPT_PRIORITY (configLIBRARY_LOWEST_INTERRUPT_PRIORITY << (8 -
    configPRIO_BITS))
00115 /* !!!! configMAX_SYSCALL_INTERRUPT_PRIORITY must not be set to zero !!!!
00116 See http://www.FreeRTOS.org/RTOS-Cortex-M3-M4.html. */
00117 #define configMAX_SYSCALL_INTERRUPT_PRIORITY (configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY << (8 -
    configPRIO_BITS))
00118
00119 /* Normal assert() semantics without relying on the provision of an assert.h
00120 header file. */
00121 #define configASSERT(x)              \
00122     if ((x) == 0)                    \
00123     {                                \
00124         taskDISABLE_INTERRUPTS(); \
00125         for (;;)                     \
00126             ;                        \
00127     }
00128
```

```
00129 /* Definitions that map the FreeRTOS port interrupt handlers to their CMSIS
00130 standard names. */
00131 #define vPortSVCHandler SVC_Handler
00132 #define xPortPendSVHandler PendSV_Handler
00133 #define xPortSysTickHandler SysTick_Handler
00134
00135 /* Functions and macros used for task statistics */
00136 extern void bspConfigureTimForRunTimeStats(void);
00137 extern uint16_t bspGetTimStatsCount(void);
00138 #define portCONFIGURE_TIMER_FOR_RUN_TIME_STATS() bspConfigureTimForRunTimeStats()
00139 #define portGET_RUN_TIME_COUNTER_VALUE() bspGetTimStatsCount();
00140
00141 #endif /* FREERTOS_CONFIG_H */
```

4.19 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.

4.19.1 Detailed Description

main header file: Holds generic handlers.

Author

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

4.19.2 Function Documentation

4.19.2.1 Error_Handler()

```
void Error_Handler (
            void )
```

This function is executed in case of error occurrence.

Return values

None	

4.20 main.h

 Go to the documentation of this file.

```
00001
00008 #include "stm32f4xx_hal.h"
00009
00010 #ifndef __MAIN_H
00011 #define __MAIN_H
00012
00013 void Error_Handler(void);
00014
00015 #endif
```

4.21 stm32f4xx_hal_conf.h

```
00001
00021 /* Define to prevent recursive inclusion -------------------------------------*/
00022 #ifndef __STM32F4xx_HAL_CONF_H
00023 #define __STM32F4xx_HAL_CONF_H
00024
00025 #ifdef __cplusplus
00026  extern "C" {
00027 #endif
00028
00029 /* Exported types ------------------------------------------------------------*/
00030 /* Exported constants --------------------------------------------------------*/
00031
00032 /* ######################### Module Selection ############################## */
00036 #define HAL_MODULE_ENABLED
00037
00038   /* #define HAL_CRYP_MODULE_ENABLED */
00039 /* #define HAL_ADC_MODULE_ENABLED */
00040 /* #define HAL_CAN_MODULE_ENABLED */
00041 /* #define HAL_CRC_MODULE_ENABLED */
00042 /* #define HAL_CAN_LEGACY_MODULE_ENABLED */
00043 /* #define HAL_DAC_MODULE_ENABLED */
00044 /* #define HAL_DCMI_MODULE_ENABLED */
00045 /* #define HAL_DMA2D_MODULE_ENABLED */
00046 /* #define HAL_ETH_MODULE_ENABLED */
00047 /* #define HAL_ETH_LEGACY_MODULE_ENABLED */
00048 /* #define HAL_NAND_MODULE_ENABLED */
00049 /* #define HAL_NOR_MODULE_ENABLED */
00050 /* #define HAL_PCCARD_MODULE_ENABLED */
00051 /* #define HAL_SRAM_MODULE_ENABLED */
00052 /* #define HAL_SDRAM_MODULE_ENABLED */
00053 /* #define HAL_HASH_MODULE_ENABLED */
00054 /* #define HAL_I2C_MODULE_ENABLED */
00055 /* #define HAL_I2S_MODULE_ENABLED */
00056 /* #define HAL_IWDG_MODULE_ENABLED */
00057 /* #define HAL_LTDC_MODULE_ENABLED */
00058 /* #define HAL_RNG_MODULE_ENABLED */
00059 /* #define HAL_RTC_MODULE_ENABLED */
00060 /* #define HAL_SAI_MODULE_ENABLED */
00061 /* #define HAL_SD_MODULE_ENABLED */
00062 /* #define HAL_MMC_MODULE_ENABLED */
00063 /* #define HAL_SPI_MODULE_ENABLED */
00064 #define HAL_TIM_MODULE_ENABLED
00065 #define HAL_UART_MODULE_ENABLED
00066 /* #define HAL_USART_MODULE_ENABLED */
00067 /* #define HAL_IRDA_MODULE_ENABLED */
00068 /* #define HAL_SMARTCARD_MODULE_ENABLED */
00069 /* #define HAL_SMBUS_MODULE_ENABLED */
00070 /* #define HAL_WWDG_MODULE_ENABLED */
00071 /* #define HAL_PCD_MODULE_ENABLED */
00072 /* #define HAL_HCD_MODULE_ENABLED */
00073 /* #define HAL_DSI_MODULE_ENABLED */
00074 /* #define HAL_QSPI_MODULE_ENABLED */
00075 /* #define HAL_QSPI_MODULE_ENABLED */
00076 /* #define HAL_CEC_MODULE_ENABLED */
00077 /* #define HAL_FMPI2C_MODULE_ENABLED */
00078 /* #define HAL_FMPSMBUS_MODULE_ENABLED */
00079 /* #define HAL_SPDIFRX_MODULE_ENABLED */
00080 /* #define HAL_DFSDM_MODULE_ENABLED */
00081 /* #define HAL_LPTIM_MODULE_ENABLED */
00082 #define HAL_GPIO_MODULE_ENABLED
00083 #define HAL_EXTI_MODULE_ENABLED
00084 #define HAL_DMA_MODULE_ENABLED
00085 #define HAL_RCC_MODULE_ENABLED
00086 #define HAL_FLASH_MODULE_ENABLED
00087 #define HAL_PWR_MODULE_ENABLED
00088 #define HAL_CORTEX_MODULE_ENABLED
00089
00090 /* ######################### HSE/HSI Values adaptation ################### */
00096 #if !defined  (HSE_VALUE)
00097   #define HSE_VALUE    25000000U
00098 #endif /* HSE_VALUE */
```

```
00099
00100 #if !defined  (HSE_STARTUP_TIMEOUT)
00101   #define HSE_STARTUP_TIMEOUT    100U
00102 #endif /* HSE_STARTUP_TIMEOUT */
00103
00109 #if !defined  (HSI_VALUE)
00110   #define HSI_VALUE     ((uint32_t)16000000U)
00111 #endif /* HSI_VALUE */
00112
00116 #if !defined  (LSI_VALUE)
00117  #define LSI_VALUE  32000U
00118 #endif /* LSI_VALUE */
00124 #if !defined  (LSE_VALUE)
00125  #define LSE_VALUE  32768U
00126 #endif /* LSE_VALUE */
00127
00128 #if !defined  (LSE_STARTUP_TIMEOUT)
00129   #define LSE_STARTUP_TIMEOUT    5000U
00130 #endif /* LSE_STARTUP_TIMEOUT */
00131
00137 #if !defined  (EXTERNAL_CLOCK_VALUE)
00138   #define EXTERNAL_CLOCK_VALUE    12288000U
00139 #endif /* EXTERNAL_CLOCK_VALUE */
00140
00141 /* Tip: To avoid modifying this file each time you need to use different HSE,
00142    ===  you can define the HSE value in your toolchain compiler preprocessor. */
00143
00144 /* ########################### System Configuration ######################### */
00148 #define  VDD_VALUE              3300U
00149 #define  TICK_INT_PRIORITY           15U
00150 #define  USE_RTOS                0U
00151 #define  PREFETCH_ENABLE            1U
00152 #define  INSTRUCTION_CACHE_ENABLE     1U
00153 #define  DATA_CACHE_ENABLE          1U
00154
00155 #define  USE_HAL_ADC_REGISTER_CALLBACKS         0U /* ADC register callback disabled       */
00156 #define  USE_HAL_CAN_REGISTER_CALLBACKS         0U /* CAN register callback disabled       */
00157 #define  USE_HAL_CEC_REGISTER_CALLBACKS         0U /* CEC register callback disabled       */
00158 #define  USE_HAL_CRYP_REGISTER_CALLBACKS        0U /* CRYP register callback disabled      */
00159 #define  USE_HAL_DAC_REGISTER_CALLBACKS         0U /* DAC register callback disabled       */
00160 #define  USE_HAL_DCMI_REGISTER_CALLBACKS        0U /* DCMI register callback disabled      */
00161 #define  USE_HAL_DFSDM_REGISTER_CALLBACKS       0U /* DFSDM register callback disabled     */
00162 #define  USE_HAL_DMA2D_REGISTER_CALLBACKS       0U /* DMA2D register callback disabled     */
00163 #define  USE_HAL_DSI_REGISTER_CALLBACKS         0U /* DSI register callback disabled       */
00164 #define  USE_HAL_ETH_REGISTER_CALLBACKS         0U /* ETH register callback disabled       */
00165 #define  USE_HAL_HASH_REGISTER_CALLBACKS        0U /* HASH register callback disabled      */
00166 #define  USE_HAL_HCD_REGISTER_CALLBACKS         0U /* HCD register callback disabled       */
00167 #define  USE_HAL_I2C_REGISTER_CALLBACKS         0U /* I2C register callback disabled       */
00168 #define  USE_HAL_FMPI2C_REGISTER_CALLBACKS      0U /* FMPI2C register callback disabled    */
00169 #define  USE_HAL_FMPSMBUS_REGISTER_CALLBACKS    0U /* FMPSMBUS register callback disabled  */
00170 #define  USE_HAL_I2S_REGISTER_CALLBACKS         0U /* I2S register callback disabled       */
00171 #define  USE_HAL_IRDA_REGISTER_CALLBACKS        0U /* IRDA register callback disabled      */
00172 #define  USE_HAL_LPTIM_REGISTER_CALLBACKS       0U /* LPTIM register callback disabled     */
00173 #define  USE_HAL_LTDC_REGISTER_CALLBACKS        0U /* LTDC register callback disabled      */
00174 #define  USE_HAL_MMC_REGISTER_CALLBACKS         0U /* MMC register callback disabled       */
00175 #define  USE_HAL_NAND_REGISTER_CALLBACKS        0U /* NAND register callback disabled      */
00176 #define  USE_HAL_NOR_REGISTER_CALLBACKS         0U /* NOR register callback disabled       */
00177 #define  USE_HAL_PCCARD_REGISTER_CALLBACKS      0U /* PCCARD register callback disabled    */
00178 #define  USE_HAL_PCD_REGISTER_CALLBACKS         0U /* PCD register callback disabled       */
00179 #define  USE_HAL_QSPI_REGISTER_CALLBACKS        0U /* QSPI register callback disabled      */
00180 #define  USE_HAL_RNG_REGISTER_CALLBACKS         0U /* RNG register callback disabled       */
00181 #define  USE_HAL_RTC_REGISTER_CALLBACKS         0U /* RTC register callback disabled       */
00182 #define  USE_HAL_SAI_REGISTER_CALLBACKS         0U /* SAI register callback disabled       */
00183 #define  USE_HAL_SD_REGISTER_CALLBACKS          0U /* SD register callback disabled        */
00184 #define  USE_HAL_SMARTCARD_REGISTER_CALLBACKS   0U /* SMARTCARD register callback disabled */
00185 #define  USE_HAL_SDRAM_REGISTER_CALLBACKS       0U /* SDRAM register callback disabled     */
00186 #define  USE_HAL_SRAM_REGISTER_CALLBACKS        0U /* SRAM register callback disabled      */
00187 #define  USE_HAL_SPDIFRX_REGISTER_CALLBACKS     0U /* SPDIFRX register callback disabled   */
00188 #define  USE_HAL_SMBUS_REGISTER_CALLBACKS       0U /* SMBUS register callback disabled     */
00189 #define  USE_HAL_SPI_REGISTER_CALLBACKS         0U /* SPI register callback disabled       */
00190 #define  USE_HAL_TIM_REGISTER_CALLBACKS         0U /* TIM register callback disabled       */
00191 #define  USE_HAL_UART_REGISTER_CALLBACKS        0U /* UART register callback disabled      */
00192 #define  USE_HAL_USART_REGISTER_CALLBACKS       0U /* USART register callback disabled     */
00193 #define  USE_HAL_WWDG_REGISTER_CALLBACKS        0U /* WWDG register callback disabled      */
00194
00195 /* ######################### Assert Selection ############################## */
00200 /* #define USE_FULL_ASSERT    1U */
00201
00202 /* ################# Ethernet peripheral configuration #################### */
00203
00204 /* Section 1 : Ethernet peripheral configuration */
00205
00206 /* MAC ADDRESS: MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */
00207 #define MAC_ADDR0   2U
00208 #define MAC_ADDR1   0U
00209 #define MAC_ADDR2   0U
00210 #define MAC_ADDR3   0U
```

```
00211 #define MAC_ADDR4    0U
00212 #define MAC_ADDR5    0U
00213
00214 /* Definition of the Ethernet driver buffers size and count */
00215 #define ETH_RX_BUF_SIZE                    /* buffer size for receive               */
00216 #define ETH_TX_BUF_SIZE                    ETH_MAX_PACKET_SIZE /* buffer size for transmit           */
00217 #define ETH_RXBUFNB                  4U        /* 4 Rx buffers of size ETH_RX_BUF_SIZE  */
00218 #define ETH_TXBUFNB                  4U        /* 4 Tx buffers of size ETH_TX_BUF_SIZE  */
00219
00220 /* Section 2: PHY configuration section */
00221
00222 /* DP83848_PHY_ADDRESS Address*/
00223 #define DP83848_PHY_ADDRESS          0x01U
00224 /* PHY Reset delay these values are based on a 1 ms Systick interrupt*/
00225 #define PHY_RESET_DELAY              0x000000FFU
00226 /* PHY Configuration delay */
00227 #define PHY_CONFIG_DELAY             0x00000FFFU
00228
00229 #define PHY_READ_TO                  0x0000FFFFU
00230 #define PHY_WRITE_TO                 0x0000FFFFU
00231
00232 /* Section 3: Common PHY Registers */
00233
00234 #define PHY_BCR                      ((uint16_t)0x0000U)
00235 #define PHY_BSR                      ((uint16_t)0x0001U)
00237 #define PHY_RESET                    ((uint16_t)0x8000U)
00238 #define PHY_LOOPBACK                 ((uint16_t)0x4000U)
00239 #define PHY_FULLDUPLEX_100M          ((uint16_t)0x2100U)
00240 #define PHY_HALFDUPLEX_100M          ((uint16_t)0x2000U)
00241 #define PHY_FULLDUPLEX_10M           ((uint16_t)0x0100U)
00242 #define PHY_HALFDUPLEX_10M           ((uint16_t)0x0000U)
00243 #define PHY_AUTONEGOTIATION          ((uint16_t)0x1000U)
00244 #define PHY_RESTART_AUTONEGOTIATION  ((uint16_t)0x0200U)
00245 #define PHY_POWERDOWN                ((uint16_t)0x0800U)
00246 #define PHY_ISOLATE                  ((uint16_t)0x0400U)
00248 #define PHY_AUTONEGO_COMPLETE        ((uint16_t)0x0020U)
00249 #define PHY_LINKED_STATUS            ((uint16_t)0x0004U)
00250 #define PHY_JABBER_DETECTION         ((uint16_t)0x0002U)
00252 /* Section 4: Extended PHY Registers */
00253 #define PHY_SR                       ((uint16_t)0x10U)
00255 #define PHY_SPEED_STATUS             ((uint16_t)0x0002U)
00256 #define PHY_DUPLEX_STATUS            ((uint16_t)0x0004U)
00258 /* ################## SPI peripheral configuration ######################## */
00259
00260 /* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver
00261 * Activated: CRC code is present inside driver
00262 * Deactivated: CRC code cleaned from driver
00263 */
00264
00265 #define USE_SPI_CRC                  0U
00266
00267 /* Includes ------------------------------------------------------------------*/
00272 #ifdef HAL_RCC_MODULE_ENABLED
00273   #include "stm32f4xx_hal_rcc.h"
00274 #endif /* HAL_RCC_MODULE_ENABLED */
00275
00276 #ifdef HAL_GPIO_MODULE_ENABLED
00277   #include "stm32f4xx_hal_gpio.h"
00278 #endif /* HAL_GPIO_MODULE_ENABLED */
00279
00280 #ifdef HAL_EXTI_MODULE_ENABLED
00281   #include "stm32f4xx_hal_exti.h"
00282 #endif /* HAL_EXTI_MODULE_ENABLED */
00283
00284 #ifdef HAL_DMA_MODULE_ENABLED
00285   #include "stm32f4xx_hal_dma.h"
00286 #endif /* HAL_DMA_MODULE_ENABLED */
00287
00288 #ifdef HAL_CORTEX_MODULE_ENABLED
00289   #include "stm32f4xx_hal_cortex.h"
00290 #endif /* HAL_CORTEX_MODULE_ENABLED */
00291
00292 #ifdef HAL_ADC_MODULE_ENABLED
00293   #include "stm32f4xx_hal_adc.h"
00294 #endif /* HAL_ADC_MODULE_ENABLED */
00295
00296 #ifdef HAL_CAN_MODULE_ENABLED
00297   #include "stm32f4xx_hal_can.h"
00298 #endif /* HAL_CAN_MODULE_ENABLED */
00299
00300 #ifdef HAL_CAN_LEGACY_MODULE_ENABLED
00301   #include "stm32f4xx_hal_can_legacy.h"
00302 #endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
00303
00304 #ifdef HAL_CRC_MODULE_ENABLED
00305   #include "stm32f4xx_hal_crc.h"
00306 #endif /* HAL_CRC_MODULE_ENABLED */
```

```
00307
00308 #ifdef HAL_CRYP_MODULE_ENABLED
00309    #include "stm32f4xx_hal_cryp.h"
00310 #endif /* HAL_CRYP_MODULE_ENABLED */
00311
00312 #ifdef HAL_DMA2D_MODULE_ENABLED
00313    #include "stm32f4xx_hal_dma2d.h"
00314 #endif /* HAL_DMA2D_MODULE_ENABLED */
00315
00316 #ifdef HAL_DAC_MODULE_ENABLED
00317    #include "stm32f4xx_hal_dac.h"
00318 #endif /* HAL_DAC_MODULE_ENABLED */
00319
00320 #ifdef HAL_DCMI_MODULE_ENABLED
00321    #include "stm32f4xx_hal_dcmi.h"
00322 #endif /* HAL_DCMI_MODULE_ENABLED */
00323
00324 #ifdef HAL_ETH_MODULE_ENABLED
00325    #include "stm32f4xx_hal_eth.h"
00326 #endif /* HAL_ETH_MODULE_ENABLED */
00327
00328 #ifdef HAL_ETH_LEGACY_MODULE_ENABLED
00329    #include "stm32f4xx_hal_eth_legacy.h"
00330 #endif /* HAL_ETH_LEGACY_MODULE_ENABLED */
00331
00332 #ifdef HAL_FLASH_MODULE_ENABLED
00333    #include "stm32f4xx_hal_flash.h"
00334 #endif /* HAL_FLASH_MODULE_ENABLED */
00335
00336 #ifdef HAL_SRAM_MODULE_ENABLED
00337    #include "stm32f4xx_hal_sram.h"
00338 #endif /* HAL_SRAM_MODULE_ENABLED */
00339
00340 #ifdef HAL_NOR_MODULE_ENABLED
00341    #include "stm32f4xx_hal_nor.h"
00342 #endif /* HAL_NOR_MODULE_ENABLED */
00343
00344 #ifdef HAL_NAND_MODULE_ENABLED
00345    #include "stm32f4xx_hal_nand.h"
00346 #endif /* HAL_NAND_MODULE_ENABLED */
00347
00348 #ifdef HAL_PCCARD_MODULE_ENABLED
00349    #include "stm32f4xx_hal_pccard.h"
00350 #endif /* HAL_PCCARD_MODULE_ENABLED */
00351
00352 #ifdef HAL_SDRAM_MODULE_ENABLED
00353    #include "stm32f4xx_hal_sdram.h"
00354 #endif /* HAL_SDRAM_MODULE_ENABLED */
00355
00356 #ifdef HAL_HASH_MODULE_ENABLED
00357  #include "stm32f4xx_hal_hash.h"
00358 #endif /* HAL_HASH_MODULE_ENABLED */
00359
00360 #ifdef HAL_I2C_MODULE_ENABLED
00361  #include "stm32f4xx_hal_i2c.h"
00362 #endif /* HAL_I2C_MODULE_ENABLED */
00363
00364 #ifdef HAL_SMBUS_MODULE_ENABLED
00365  #include "stm32f4xx_hal_smbus.h"
00366 #endif /* HAL_SMBUS_MODULE_ENABLED */
00367
00368 #ifdef HAL_I2S_MODULE_ENABLED
00369  #include "stm32f4xx_hal_i2s.h"
00370 #endif /* HAL_I2S_MODULE_ENABLED */
00371
00372 #ifdef HAL_IWDG_MODULE_ENABLED
00373  #include "stm32f4xx_hal_iwdg.h"
00374 #endif /* HAL_IWDG_MODULE_ENABLED */
00375
00376 #ifdef HAL_LTDC_MODULE_ENABLED
00377  #include "stm32f4xx_hal_ltdc.h"
00378 #endif /* HAL_LTDC_MODULE_ENABLED */
00379
00380 #ifdef HAL_PWR_MODULE_ENABLED
00381  #include "stm32f4xx_hal_pwr.h"
00382 #endif /* HAL_PWR_MODULE_ENABLED */
00383
00384 #ifdef HAL_RNG_MODULE_ENABLED
00385  #include "stm32f4xx_hal_rng.h"
00386 #endif /* HAL_RNG_MODULE_ENABLED */
00387
00388 #ifdef HAL_RTC_MODULE_ENABLED
00389  #include "stm32f4xx_hal_rtc.h"
00390 #endif /* HAL_RTC_MODULE_ENABLED */
00391
00392 #ifdef HAL_SAI_MODULE_ENABLED
00393  #include "stm32f4xx_hal_sai.h"
```

```
00394 #endif /* HAL_SAI_MODULE_ENABLED */
00395
00396 #ifdef HAL_SD_MODULE_ENABLED
00397  #include "stm32f4xx_hal_sd.h"
00398 #endif /* HAL_SD_MODULE_ENABLED */
00399
00400 #ifdef HAL_SPI_MODULE_ENABLED
00401  #include "stm32f4xx_hal_spi.h"
00402 #endif /* HAL_SPI_MODULE_ENABLED */
00403
00404 #ifdef HAL_TIM_MODULE_ENABLED
00405  #include "stm32f4xx_hal_tim.h"
00406 #endif /* HAL_TIM_MODULE_ENABLED */
00407
00408 #ifdef HAL_UART_MODULE_ENABLED
00409  #include "stm32f4xx_hal_uart.h"
00410 #endif /* HAL_UART_MODULE_ENABLED */
00411
00412 #ifdef HAL_USART_MODULE_ENABLED
00413  #include "stm32f4xx_hal_usart.h"
00414 #endif /* HAL_USART_MODULE_ENABLED */
00415
00416 #ifdef HAL_IRDA_MODULE_ENABLED
00417  #include "stm32f4xx_hal_irda.h"
00418 #endif /* HAL_IRDA_MODULE_ENABLED */
00419
00420 #ifdef HAL_SMARTCARD_MODULE_ENABLED
00421  #include "stm32f4xx_hal_smartcard.h"
00422 #endif /* HAL_SMARTCARD_MODULE_ENABLED */
00423
00424 #ifdef HAL_WWDG_MODULE_ENABLED
00425  #include "stm32f4xx_hal_wwdg.h"
00426 #endif /* HAL_WWDG_MODULE_ENABLED */
00427
00428 #ifdef HAL_PCD_MODULE_ENABLED
00429  #include "stm32f4xx_hal_pcd.h"
00430 #endif /* HAL_PCD_MODULE_ENABLED */
00431
00432 #ifdef HAL_HCD_MODULE_ENABLED
00433  #include "stm32f4xx_hal_hcd.h"
00434 #endif /* HAL_HCD_MODULE_ENABLED */
00435
00436 #ifdef HAL_DSI_MODULE_ENABLED
00437  #include "stm32f4xx_hal_dsi.h"
00438 #endif /* HAL_DSI_MODULE_ENABLED */
00439
00440 #ifdef HAL_QSPI_MODULE_ENABLED
00441  #include "stm32f4xx_hal_qspi.h"
00442 #endif /* HAL_QSPI_MODULE_ENABLED */
00443
00444 #ifdef HAL_CEC_MODULE_ENABLED
00445  #include "stm32f4xx_hal_cec.h"
00446 #endif /* HAL_CEC_MODULE_ENABLED */
00447
00448 #ifdef HAL_FMPI2C_MODULE_ENABLED
00449  #include "stm32f4xx_hal_fmpi2c.h"
00450 #endif /* HAL_FMPI2C_MODULE_ENABLED */
00451
00452 #ifdef HAL_FMPSMBUS_MODULE_ENABLED
00453  #include "stm32f4xx_hal_fmpsmbus.h"
00454 #endif /* HAL_FMPSMBUS_MODULE_ENABLED */
00455
00456 #ifdef HAL_SPDIFRX_MODULE_ENABLED
00457  #include "stm32f4xx_hal_spdifrx.h"
00458 #endif /* HAL_SPDIFRX_MODULE_ENABLED */
00459
00460 #ifdef HAL_DFSDM_MODULE_ENABLED
00461  #include "stm32f4xx_hal_dfsdm.h"
00462 #endif /* HAL_DFSDM_MODULE_ENABLED */
00463
00464 #ifdef HAL_LPTIM_MODULE_ENABLED
00465  #include "stm32f4xx_hal_lptim.h"
00466 #endif /* HAL_LPTIM_MODULE_ENABLED */
00467
00468 #ifdef HAL_MMC_MODULE_ENABLED
00469  #include "stm32f4xx_hal_mmc.h"
00470 #endif /* HAL_MMC_MODULE_ENABLED */
00471
00472 /* Exported macro -----------------------------------------------------------*/
00473 #ifdef  USE_FULL_ASSERT
00482  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
00483 /* Exported functions ------------------------------------------------------- */
00484  void assert_failed(uint8_t* file, uint32_t line);
00485 #else
00486  #define assert_param(expr) ((void)0U)
00487 #endif /* USE_FULL_ASSERT */
00488
```

```
00489 #ifdef __cplusplus
00490 }
00491 #endif
00492
00493 #endif /* __STM32F4xx_HAL_CONF_H */
```

4.22 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.

4.22.1 Detailed Description

Interrupt header file: Holds interrupt handlers.

Author

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

4.23 stm32f4xx_it.h

Go to the documentation of this file.
```
00001
00009 #ifndef __STM32F4xx_IT_H
00010 #define __STM32F4xx_IT_H
00011
00012 void NMI_Handler(void);
00013 void HardFault_Handler(void);
00014 void MemManage_Handler(void);
00015 void BusFault_Handler(void);
00016 void UsageFault_Handler(void);
00017 void DebugMon_Handler(void);
00018 void TIM1_BRK_TIM9_IRQHandler(void);
00019
00020 #endif
```

4.24 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 **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

- static BaseType_t **prvCommandPwmSetFreq** (char ∗pcWriteBuffer, size_t xWriteBufferLen, const char ∗pcCommandString)

    Command that sets a new pwm frequency.
- static BaseType_t **prvCommandPwmSetDuty** (char ∗pcWriteBuffer, size_t xWriteBufferLen, const char ∗pcCommandString)

    Command that sets a new pwm duty cycle.
- static BaseType_t **prvCommandGpioWrite** (char ∗pcWriteBuffer, size_t xWriteBufferLen, const char ∗pc↩ CommandString)

    Command that writes to a GPIOx pin.
- static BaseType_t **prvCommandGpioRead** (char ∗pcWriteBuffer, size_t xWriteBufferLen, const char ∗pc↩ CommandString)

    Command that reads from GPIOx Pin.
- static BaseType_t **prvCommandEcho** (char ∗pcWriteBuffer, size_t xWriteBufferLen, const char ∗pc↩ CommandString)

    Echo command line in UNIX systems.
- static BaseType_t **prvCommandTaskStats** (char ∗pcWriteBuffer, size_t xWriteBufferLen, const char ∗pc↩ CommandString)

    Command that gets task statistics.
- static BaseType_t **prvCommandHeap** (char ∗pcWriteBuffer, size_t xWriteBufferLen, const char ∗pc↩ CommandString)

    Command that gets heap information.
- static BaseType_t **prvCommandClk** (char ∗pcWriteBuffer, size_t xWriteBufferLen, const char ∗pc↩ CommandString)

    Command that gets clock information from current clock settings.
- static BaseType_t **prvCommandTicks** (char ∗pcWriteBuffer, size_t xWriteBufferLen, const char ∗pc↩ CommandString)

    Command that calculate OS ticks information.
- static const char ∗ **prvpcMapTaskState** (eTaskState eState)

    This function is executed in case of error occurrence.
- static BaseType_t **xConsoleRead** (uint8_t ∗cReadChar, size_t xLen)

Reads from UART RX buffer. Reads one bye at the time.
- static HAL_StatusTypeDef **vConsoleWrite** (const char ∗buff)

    Write to UART TX.
- void **vConsoleEnableRxInterrupt** (void)

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

    Task to handle user commands via serial communication.
- BaseType_t **xConsoleInit** (uint16_t usStackSize, UBaseType_t uxPriority, UART_HandleTypeDef ∗pxUart↩
Handle)

    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**
- static const char ∗ **pcWelcomeMsg** = "Welcome to the console. Enter 'help' to view a list of available commands.\n"
- static const char ∗ **prvpcTaskListHeader**
- static const char ∗ **prvpcPrompt** = "#cmd: "
- static const CLI_Command_Definition_t **xCommands** [ ]

4.24.1 Detailed Description

Command Line Interpreter based on FreeRTOS and STM32 HAL layer.

Author

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

4.24.2 Function Documentation

4.24.2.1 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	

4.24.2.2 prvCommandClk()

```
static BaseType_t prvCommandClk (
            char * pcWriteBuffer,
            size_t xWriteBufferLen,
            const char * pcCommandString ) [static]
```

Command that gets clock information from current clock settings.

Parameters

*pcWriteBuffer	FreeRTOS CLI write buffer.
xWriteBufferLen	Length of write buffer.
*pcCommandString	pointer to the command name.

Return values

HAL	status

4.24.2.3 prvCommandEcho()

```
static BaseType_t prvCommandEcho (
            char * pcWriteBuffer,
            size_t xWriteBufferLen,
            const char * pcCommandString ) [static]
```

Echo command line in UNIX systems.

Parameters

*pcWriteBuffer	FreeRTOS CLI write buffer.
xWriteBufferLen	Length of write buffer.
*pcCommandString	pointer to the command name.

Return values

HAL	status

4.24.2.4 prvCommandGpioRead()

```
static BaseType_t prvCommandGpioRead (
            char * pcWriteBuffer,
            size_t xWriteBufferLen,
            const char * pcCommandString ) [static]
```

Command that reads from GPIOx Pin.

Parameters

*pcWriteBuffer	FreeRTOS CLI write buffer.
xWriteBufferLen	Length of write buffer.
*pcCommandString	pointer to the command name.

Return values

HAL	status

4.24.2.5 prvCommandGpioWrite()

```
static BaseType_t prvCommandGpioWrite (
            char * pcWriteBuffer,
            size_t xWriteBufferLen,
            const char * pcCommandString ) [static]
```

Command that writes to a GPIOx pin.

Parameters

*pcWriteBuffer	FreeRTOS CLI write buffer.
xWriteBufferLen	Length of write buffer.
*pcCommandString	pointer to the command name.

Return values

HAL	status

4.24.2.6 prvCommandHeap()

```
static BaseType_t prvCommandHeap (
            char * pcWriteBuffer,
            size_t xWriteBufferLen,
            const char * pcCommandString ) [static]
```

Command that gets heap information.

Parameters

*pcWriteBuffer	FreeRTOS CLI write buffer.
xWriteBufferLen	Length of write buffer.
*pcCommandString	pointer to the command name.

Return values

HAL	status

4.24.2.7 prvCommandPwmSetDuty()

```
static BaseType_t prvCommandPwmSetDuty (
            char * pcWriteBuffer,
            size_t xWriteBufferLen,
            const char * pcCommandString ) [static]
```

Command that sets a new pwm duty cycle.

Parameters

∗pcWriteBuffer	FreeRTOS CLI write buffer.
xWriteBufferLen	Length of write buffer.
∗pcCommandString	pointer to the command name.

Return values

HAL	status

4.24.2.8 prvCommandPwmSetFreq()

```
static BaseType_t prvCommandPwmSetFreq (
            char * pcWriteBuffer,
            size_t xWriteBufferLen,
            const char * pcCommandString ) [static]
```

Command that sets a new pwm frequency.

Parameters

∗pcWriteBuffer	FreeRTOS CLI write buffer.
xWriteBufferLen	Length of write buffer.
∗pcCommandString	pointer to the command name.

Return values

HAL	status

4.24.2.9 prvCommandTaskStats()

```
static BaseType_t prvCommandTaskStats (
            char * pcWriteBuffer,
            size_t xWriteBufferLen,
            const char * pcCommandString ) [static]
```

Command that gets task statistics.

Parameters

*pcWriteBuffer	FreeRTOS CLI write buffer.
xWriteBufferLen	Length of write buffer.
*pcCommandString	pointer to the command name.

Return values

HAL	status

4.24.2.10 prvCommandTicks()

```
static BaseType_t prvCommandTicks (
            char * pcWriteBuffer,
            size_t xWriteBufferLen,
            const char * pcCommandString ) [static]
```

Command that calculate OS ticks information.

Parameters

*pcWriteBuffer	FreeRTOS CLI write buffer.
xWriteBufferLen	Length of write buffer.
*pcCommandString	pointer to the command name.

Return values

HAL	status

4.24.2.11 prvpcMapTaskState()

```
static const char * prvpcMapTaskState (
            eTaskState eState ) [static]
```

This function is executed in case of error occurrence.

Return values

None	

4.24.2.12 vConsoleEnableRxInterrupt()

```
void vConsoleEnableRxInterrupt (
            void  )
```

Enables UART RX reception.

Parameters

void	

Return values

void	

4.24.2.13 vConsoleWrite()

```
static HAL_StatusTypeDef vConsoleWrite (
            const char * buff )  [static]
```

Write to UART TX.

Parameters

∗buff	buffer to be written.

Return values

HAL	status

4.24.2.14 vTaskConsole()

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

Task to handle user commands via serial communication.

Parameters

∗pvParams	Data passed at task creation.

Return values

void	

4.24.2.15 xConsoleInit()

```
BaseType_t xConsoleInit (
            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

4.24.2.16 xConsoleRead()

```
static BaseType_t xConsoleRead (
            uint8_t * cReadChar,
            size_t xLen )  [static]
```

Reads from UART RX buffer. Reads one bye at the time.

Parameters

∗*cReadChar*	pointer to where data will be stored.

Return values

HAL	status

4.24.3 Variable Documentation

4.24.3.1 prvpcTaskListHeader

```
const char* prvpcTaskListHeader  [static]
```

Initial value:
```
= "Task states: Bl = Blocked, Re = Ready, Ru = Running, De = Deleted,  Su = Suspended\n\n"
                                "Task name          State  Priority  Stack remaining  CPU usage
     Runtime(us)\n"
                                "================= =====  ========  ===============  =========
     ==========\n"
```

4.25 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**

4.25.1   Detailed Description

Command Line Interpreter based on FreeRTOS and STM32 HAL layer.

Author

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

4.25.2   Function Documentation

4.25.2.1   Error_Handler()

```
void Error_Handler (
            void  )
```

This function is executed in case of error occurrence.

Return values

None	

4.25.2.2   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	

4.25.2.3 main()

```
int main (
            void )
```

main function: Initialize BSP, console, and FreeRTOS.

Parameters

void	

Return values

void	

4.25.2.4 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	

4.26 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.

4.26.1 Detailed Description

Interrupt header file: Holds interrupt handlers.

Author

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

4.26.2 Function Documentation

4.26.2.1 HAL_MspInit()

```
void HAL_MspInit (
            void  )
```

Enable peripheral clocks and set NVIC priorities.

Parameters

void	

Return values

void	

4.26.2.2 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	

4.26.2.3 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	

4.27 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**

4.27.1 Detailed Description

HAL time base based on the hardware TIM.

Attention

Copyright (c) 2024 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

4.27.2 Function Documentation

4.27.2.1 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

4.27.2.2 HAL_ResumeTick()

```
void HAL_ResumeTick (
            void )
```

Resume Tick increment.

Note

> Enable the tick increment by Enabling TIM9 update interrupt.

Parameters

None	

Return values

None	

4.27.2.3 HAL_SuspendTick()

```
void HAL_SuspendTick (
            void )
```

Suspend Tick increment.

Note

Disable the tick increment by disabling TIM9 update interrupt.

Parameters

None	

Return values

None	

4.28 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**

4.28.1 Detailed Description

Interrupt source file: Holds interrupt handler implementations.

Author

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

4.29 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

4.29.1 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

4.30 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() (p. *52*) allocates memory to the newlib heap and is used by malloc and others from the C library

Variables

- static uint8_t ∗ **__sbrk_heap_end** = NULL

4.30.1 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

4.30.2 Function Documentation

4.30.2.1 _sbrk()

```
void ∗ _sbrk (
            ptrdiff_t incr )
```

_sbrk() (p. 52) 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

4.30.3   Variable Documentation

4.30.3.1   __sbrk_heap_end

```
uint8_t* __sbrk_heap_end = NULL  [static]
```

Pointer to the current high watermark of the heap usage

4.31   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}

4.31.1 Detailed Description

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

Author

MCD Application Team

This file provides two functions and one global variable to be called from user application:

- **SystemInit()** (p. 7): This function is called at startup just after reset and before branch to main program. This call is made inside the "startup_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()** (p. 7): Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.

Attention

Index