Controle de temperatura

Gerado por Doxygen 1.9.2

1 Índice dos Módulos
1.1 Módulos
2 Índice dos Arquivos
2.1 Lista de Arquivos
3 Módulos
3.1 CMSIS
3.1.1 Descrição detalhada
3.2 Stm32f1xx_system
3.2.1 Descrição detalhada
3.3 STM32F1xx_System_Private_Includes
3.4 STM32F1xx_System_Private_TypesDefinitions
3.5 STM32F1xx_System_Private_Defines
3.5.1 Descrição detalhada
3.5.2 Definições e macros
3.5.2.1 HSE_VALUE
3.5.2.2 HSI_VALUE
3.6 STM32F1xx_System_Private_Macros
3.7 STM32F1xx_System_Private_Variables
3.7.1 Descrição detalhada
3.7.2 Variáveis
3.7.2.1 AHBPrescTable
3.7.2.2 APBPrescTable
3.7.2.3 SystemCoreClock
3.8 STM32F1xx_System_Private_FunctionPrototypes
3.9 STM32F1xx_System_Private_Functions
3.9.1 Descrição detalhada
3.9.2 Funções
3.9.2.1 SystemCoreClockUpdate()
3.9.2.2 SystemInit()
o.o.e.e oyotommit()
4 Arquivos
4.1 Referência do Arquivo D:/BACKUP/Faculdade/16_Embarcados/Controlador_Temperatura/Controlede-temperatura/Codigos/controle_temperatura_RTOS/Core/Src/FIRFilter.c
4.1.1 Descrição detalhada
4.1.2 Funções
4.1.2.1 FIRFilter_Init()
4.1.2.2 FIRFilter_Update()
4.2 FIRFilter.c
4.3 freertos.c
4.4 Referência do Arquivo D:/BACKUP/Faculdade/16 Embarcados/Controlador Temperatura/Controle-
de-temperatura/Codigos/controle_temperatura_RTOS/Core/Src/main.c
4.4.1 Descrição detalhada

4.	4.2 Definições e macros	16
	4.4.2.1 CSdis	16
	4.4.2.2 CSen	16
	4.4.2.3 k	16
	4.4.2.4 ksat	16
	4.4.2.5 p	17
	4.4.2.6 r	17
	4.4.2.7 SCK_H	17
	4.4.2.8 SCK_L	17
4.	4.3 Funções	17
	4.4.3.1 Control_taskF()	17
	4.4.3.2 Display_taskF()	18
	4.4.3.3 Error_Handler()	19
	4.4.3.4 Filter_taskF()	20
	4.4.3.5 HAL_TIM_PeriodElapsedCallback()	20
		21
	4.4.3.7 SystemClock_Config()	22
	4.4.3.8 Temp_taskF()	22
4.	4.4 Variáveis	23
		23
	4.4.4.2 Display_Task	24
	4.4.4.3 dutyCycle	24
	<del>-</del>	24
	•	24
	4.4.4.6 filteredTempQueue	24
	4.4.4.7 hadc1	24
	4.4.4.8 htim1	25
		25
		25
	· <del>-</del>	25
	•	25
	•	25
		26
	erência do Arquivo D:/BACKUP/Faculdade/16_Embarcados/Controlador_Temperatura/Controle-e-temperatura/Codigos/controle_temperatura_RTOS/Core/Src/stm32f1xx_hal_msp.c	31
		31
		31
٠.		31
		32
		33
		33
		34

4.6.2.6 HAL_TIM_PWM_MspInit()	34
4.6.2.7 HAL_UART_MspDeInit()	35
4.6.2.8 HAL_UART_MspInit()	35
4.7 stm32f1xx_hal_msp.c	36
4.8 Referência do Arquivo D:/BACKUP/Faculdade/16_Embarcados/Controlador_Temperatura/Controlede-temperatura/Codigos/controle_temperatura_RTOS/Core/Src/stm32f1xx_hal_timebase_tim.c	39
4.8.1 Descrição detalhada	39
4.8.2 Funções	39
4.8.2.1 HAL_InitTick()	40
4.8.2.2 HAL_ResumeTick()	41
4.8.2.3 HAL_SuspendTick()	41
4.8.3 Variáveis	42
4.8.3.1 htim4	42
4.9 stm32f1xx_hal_timebase_tim.c	42
4.10 Referência do Arquivo D:/BACKUP/Faculdade/16_Embarcados/Controlador_Temperatura/Controle-de-temperatura/Codigos/controle_temperatura_RTOS/Core/Src/stm32f1xx_it.c	43
4.10.1 Descrição detalhada	43
4.10.2 Funções	44
4.10.2.1 BusFault_Handler()	44
4.10.2.2 DebugMon_Handler()	44
4.10.2.3 HardFault_Handler()	45
4.10.2.4 MemManage_Handler()	45
4.10.2.5 NMI_Handler()	45
4.10.2.6 TIM4_IRQHandler()	46
4.10.2.7 UsageFault_Handler()	46
4.10.3 Variáveis	46
4.10.3.1 htim4	46
4.11 stm32f1xx_it.c	47
4.12 Referência do Arquivo D:/BACKUP/Faculdade/16_Embarcados/Controlador_Temperatura/Controle-de-temperatura/Codigos/controle_temperatura_RTOS/Core/Src/syscalls.c	48
4.12.1 Descrição detalhada	49
4.12.2 Funções	50
4.12.2.1attribute()	50
4.12.2.2io_getchar()	50
4.12.2.3 _close()	50
4.12.2.4 _execve()	50
4.12.2.5 _exit()	51
4.12.2.6 _fork()	51
4.12.2.7 _fstat()	51
4.12.2.8 _getpid()	51
4.12.2.9 _isatty()	52
4.12.2.10 _kill()	52
4.12.2.11 _link()	52

4.12.2.12 _lseek()	52
4.12.2.13 _open()	53
4.12.2.14 _stat()	53
4.12.2.15 _times()	53
4.12.2.16 _unlink()	53
4.12.2.17 _wait()	54
4.12.2.18 initialise_monitor_handles()	54
4.12.3 Variáveis	54
4.12.3.1 environ	54
4.13 syscalls.c	54
4.14 Referência do Arquivo D:/BACKUP/Faculdade/16_Embarcados/Controlador_Temperatura/Controle-de-temperatura/Codigos/controle_temperatura_RTOS/Core/Src/sysmem.c	56
4.14.1 Descrição detalhada	56
4.14.2 Funções	57
4.14.2.1 _sbrk()	57
4.15 sysmem.c	58
4.16 Referência do Arquivo D:/BACKUP/Faculdade/16_Embarcados/Controlador_Temperatura/Controle-de-temperatura/Codigos/controle_temperatura_RTOS/Core/Src/system_stm32f1xx.c	58
4.16.1 Descrição detalhada	59
4.17 system_stm32f1xx.c	60
Índice Remissivo	63

## Capítulo 1

## **Índice dos Módulos**

#### 1.1 Módulos

Esta é a lista de todos os módulos:

CMSIS
Stm32f1xx_system
STM32F1xx_System_Private_Includes
STM32F1xx_System_Private_TypesDefinitions
STM32F1xx_System_Private_Defines
STM32F1xx_System_Private_Macros
STM32F1xx_System_Private_Variables
STM32F1xx_System_Private_FunctionPrototypes
STM32F1xx System Private Functions

2 Índice dos Módulos

## Capítulo 2

# Índice dos Arquivos

## 2.1 Lista de Arquivos

Esta é a lista de todos os arquivos documentados e suas respectivas descrições:

$D:/BACKUP/Faculdade/16\_Embarcados/Controlador\_Temperatura/Controle-de-temperatura/Codigos/controle \leftarrow 1.00 + 1.00$
_temperatura_RTOS/Core/Src/FIRFilter.c
Implementação do filtro FIR
D:/BACKUP/Faculdade/16_Embarcados/Controlador_Temperatura/Controle-de-temperatura/Codigos/controle-
_temperatura_RTOS/Core/Src/freertos.c
D:/BACKUP/Faculdade/16_Embarcados/Controlador_Temperatura/Controle-de-temperatura/Codigos/controle-
_temperatura_RTOS/Core/Src/main.c
Corpo principal do programa
D:/BACKUP/Faculdade/16_Embarcados/Controlador_Temperatura/Controle-de-temperatura/Codigos/controle-
_temperatura_RTOS/Core/Src/stm32f1xx_hal_msp.c
This file provides code for the MSP Initialization and de-Initialization codes
D:/BACKUP/Faculdade/16_Embarcados/Controlador_Temperatura/Controle-de-temperatura/Codigos/controle-
_temperatura_RTOS/Core/Src/stm32f1xx_hal_timebase_tim.c
HAL time base based on the hardware TIM
$D: BACKUP/Faculdade/16\_Embarcados/Controlador\_Temperatura/Controle-de-temperatura/Codigos/controle \leftarrow Codigos/Controlador\_Temperatura/Controle-de-temperatura/Codigos/Controlador\_Temperatura/Controle-de-temperatura/Codigos/Controlador\_Temperatura/Codigos/Codigos/Controlador\_Temperatura/Codigos$
_temperatura_RTOS/Core/Src/stm32f1xx_it.c
Interrupt Service Routines
$D:/BACKUP/Faculdade/16\_Embarcados/Controlador\_Temperatura/Controle-de-temperatura/Codigos/controle \leftarrow Codigos/Controlador\_Temperatura/Controle-de-temperatura/Codigos/Controlador\_Temperatura/Controle-de-temperatura/Codigos/Controlador\_Temperatura/Codigos/Codigos/Controlador\_Temperatura/Codigos$
_temperatura_RTOS/Core/Src/syscalls.c
STM32CubeIDE Minimal System calls file
$D:/BACKUP/Faculdade/16\_Embarcados/Controlador\_Temperatura/Controle-de-temperatura/Codigos/controle \leftarrow Codigos/Controlador\_Temperatura/Controle-de-temperatura/Codigos/Controlador\_Temperatura/Controle-de-temperatura/Codigos/Controlador\_Temperatura/Codigos/Codigos/Controlador\_Temperatura/Codigos$
_temperatura_RTOS/Core/Src/sysmem.c
STM32CubeIDE System Memory calls file
$D:/BACKUP/Faculdade/16\_Embarcados/Controlador\_Temperatura/Controle-de-temperatura/Codigos/controle \leftarrow Codigos/Controlador\_Temperatura/Controle-de-temperatura/Codigos/Controlador\_Temperatura/Controle-de-temperatura/Codigos/Controlador\_Temperatura/Codigos/Codigos/Controlador\_Temperatura/Codigos$
_temperatura_RTOS/Core/Src/system_stm32f1xx.c
CMSIS Cortex-M3 Device Peripheral Access Laver System Source File

Índice dos Arquivos

## Capítulo 3

## Módulos

#### 3.1 CMSIS

#### **Módulos**

- Stm32f1xx\_system
- 3.1.1 Descrição detalhada
- 3.2 Stm32f1xx\_system

#### Módulos

- STM32F1xx\_System\_Private\_Includes
- STM32F1xx\_System\_Private\_TypesDefinitions
- STM32F1xx\_System\_Private\_Defines
- STM32F1xx\_System\_Private\_Macros
- STM32F1xx\_System\_Private\_Variables
- STM32F1xx\_System\_Private\_FunctionPrototypes
- STM32F1xx\_System\_Private\_Functions
- 3.2.1 Descrição detalhada
- 3.3 STM32F1xx\_System\_Private\_Includes
- 3.4 STM32F1xx\_System\_Private\_TypesDefinitions
- 3.5 STM32F1xx\_System\_Private\_Defines

#### Definições e Macros

- #define HSE\_VALUE 8000000U
- #define HSI\_VALUE 8000000U

6 Módulos

#### 3.5.1 Descrição detalhada

#### 3.5.2 Definições e macros

#### 3.5.2.1 HSE\_VALUE

```
#define HSE_VALUE 8000000U
```

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

Definição na linha 79 do arquivo system\_stm32f1xx.c.

#### 3.5.2.2 HSI\_VALUE

```
#define HSI_VALUE 8000000U
```

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

Definição na linha 84 do arquivo system\_stm32f1xx.c.

#### 3.6 STM32F1xx\_System\_Private\_Macros

#### 3.7 STM32F1xx\_System\_Private\_Variables

#### **Variáveis**

- uint32\_t SystemCoreClock = 16000000
- const uint8\_t AHBPrescTable [16U] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8\_t APBPrescTable [8U] = {0, 0, 0, 0, 1, 2, 3, 4}

#### 3.7.1 Descrição detalhada

#### 3.7.2 Variáveis

#### 3.7.2.1 AHBPrescTable

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

Definição na linha 143 do arquivo system\_stm32f1xx.c.

#### 3.7.2.2 APBPrescTable

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

Definição na linha 144 do arquivo system stm32f1xx.c.

#### 3.7.2.3 SystemCoreClock

```
uint32_t SystemCoreClock = 16000000
```

Definição na linha 142 do arquivo system\_stm32f1xx.c.

#### 3.8 STM32F1xx\_System\_Private\_FunctionPrototypes

#### 3.9 STM32F1xx\_System\_Private\_Functions

#### **Funções**

void SystemInit (void)

Setup the microcontroller system Initialize the Embedded Flash Interface, the PLL and update the SystemCoreClock variable.

void SystemCoreClockUpdate (void)

Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

#### 3.9.1 Descrição detalhada

#### 3.9.2 Funções

#### 3.9.2.1 SystemCoreClockUpdate()

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.

8 Módulos

#### Observação

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 by the PLL factors.
- (\*) HSI\_VALUE is a constant defined in stm32f1xx.h file (default value 8 MHz) but the real value may vary depending on the variations in voltage and temperature.
- (\*\*) HSE\_VALUE is a constant defined in stm32f1xx.h file (default value 8 MHz or 25 MHz, depending on the product used), 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.

#### **Parâmetros**

None

#### Valores Retornados

None

#### Definição na linha 225 do arquivo system stm32f1xx.c.

```
00226
00227
        uint32_t tmp = 0U, pllmul1 = 0U, pllsource = 0U;
00228
00229 #if defined(STM32F105xC) || defined(STM32F107xC)
00230
        uint32_t prediv1source = 0U, prediv1factor = 0U, prediv2factor = 0U, pl12mul1 = 0U;
00231 #endif /* STM32F105xC */
00232
00233 #if defined(STM32F100xB) || defined(STM32F100xE)
00234 uint32_t prediv1factor = 0U;
00235 #endif /* STM32F100xB or STM32F100xE */
00236
00237
        /* Get SYSCLK source --
00238
        tmp = RCC->CFGR & RCC_CFGR_SWS;
00239
00240
        switch (tmp)
00241
00242
          case 0x00U: /* HSI used as system clock */
00243
            SystemCoreClock = HSI_VALUE;
            break;
00244
          case 0x04U:
                       /* HSE used as system clock */
00245
            SystemCoreClock = HSE_VALUE;
00246
00247
00248
          case 0x08U: /* PLL used as system clock */
00249
00250
            /* Get PLL clock source and multiplication factor -----*/
            pllmull = RCC->CFGR & RCC_CFGR_PLLMULL;
00251
            pllsource = RCC->CFGR & RCC_CFGR_PLLSRC;
00252
00254 #if !defined(STM32F105xC) && !defined(STM32F107xC)
```

```
00255
             pllmull = ( pllmull » 18U) + 2U;
00256
00257
              if (pllsource == 0x00U)
00258
               /* HSI oscillator clock divided by 2 selected as PLL clock entry */ SystemCoreClock = (HSI_VALUE » 1U) * pllmull;
00259
00260
00261
00262
00263
        #if defined(STM32F100xB) || defined(STM32F100xE)
    predivlfactor = (RCC->CFGR2 & RCC_CFGR2_PREDIV1) + 1U;
00264
00265
               /* HSE oscillator clock selected as PREDIV1 clock entry */
00266
00267
               SystemCoreClock = (HSE_VALUE / prediv1factor) * pllmull;
00268 #else
00269
                /\star HSE selected as PLL clock entry \star/
                if ((RCC->CFGR & RCC_CFGR_PLLXTPRE) != (uint32_t)RESET) \{/\star \text{ HSE oscillator clock divided by 2 }\star/
00270
00271
00272
                  SystemCoreClock = (HSE_VALUE » 1U) * pllmull;
00273
00274
                else
00275
                {
00276
                  SystemCoreClock = HSE_VALUE * pllmull;
00277
00278 #endif
00279
00280 #else
00281
              pllmull = pllmull » 18U;
00282
00283
              if (pllmull != 0x0DU)
00284
                 pllmull += 2U;
00285
00286
00287
00288
              { /* PLL multiplication factor = PLL input clock * 6.5 */
00289
               pllmull = 13U / 2U;
00290
00291
00292
              if (pllsource == 0x00U)
00293
              {
00294
                /\star HSI oscillator clock divided by 2 selected as PLL clock entry \star/
00295
                SystemCoreClock = (HSI_VALUE » 1U) * pllmull;
00296
00297
              else
00298
              {/* PREDIV1 selected as PLL clock entry */
00299
00300
                /\star Get PREDIV1 clock source and division factor \star/
                prediv1source = RCC->CFGR2 & RCC_CFGR2_PREDIV1SRC;
prediv1factor = (RCC->CFGR2 & RCC_CFGR2_PREDIV1) + 1U;
00301
00302
00303
00304
                if (prediv1source == 0U)
00305
                {
00306
                   /\star HSE oscillator clock selected as PREDIV1 clock entry \star/
00307
                  SystemCoreClock = (HSE_VALUE / prediv1factor) * pllmull;
00308
00309
                else
00310
                {/* PLL2 clock selected as PREDIV1 clock entry */
00311
00312
                   /* Get PREDIV2 division factor and PLL2 multiplication factor */
                  prediv2factor = ((RCC->CFGR2 & RCC_CFGR2_PREDIV2) » 4U) + 1U;
pll2mull = ((RCC->CFGR2 & RCC_CFGR2_PLL2MUL) » 8U) + 2U;
SystemCoreClock = (((HSE_VALUE / prediv2factor) * pll2mull) / prediv1factor) * pllmull;
00313
00314
00315
00316
                }
00317
00318 #endif /* STM32F105xC */
00319
           break;
00320
00321
           default:
00322
             SystemCoreClock = HSI_VALUE;
00323
              break;
00324 }
00325
00326
         /* Compute HCLK clock frequency ----*/
         /* Get HCLK prescaler */
00327
         tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) » 4U)];
00328
         /* HCLK clock frequency */
00329
00330
         SystemCoreClock >= tmp;
00331 }
```

#### 3.9.2.2 SystemInit()

```
void SystemInit (
```

10 Módulos

```
void )
```

Setup the microcontroller system Initialize the Embedded Flash Interface, the PLL and update the SystemCoreClock variable.

#### Observação

This function should be used only after reset.

#### **Parâmetros**

None

#### Valores Retornados

None

#### Definição na linha 176 do arquivo system\_stm32f1xx.c.

```
00177 {
00178 #if defined(STM32F100xE) || defined(STM32F101xE) || defined(STM32F101xG) || defined(STM32F103xE) || defined(STM32F103xB) || defined(STM32F103xB) || defined(STM32F103xB) || defined(STM32F103xB) || defined(STM32F103xB) || defined(STM32F101xB) || defined(STM3
```

## Capítulo 4

## **Arquivos**

4.1 Referência do Arquivo D:/BACKUP/Faculdade/16\_Embarcados/

Controlador\_Temperatura/Controle-de-temperatura/

Codigos/controle temperatura RTOS/Core/Src/FIRFilter.c

```
Implementação do filtro FIR.
#include "FIRFilter.h"
```

#### **Funções**

- void FIRFilter\_Init (FIRFilter \*fir)
- float FIRFilter\_Update (FIRFilter \*fir, float inp)

#### 4.1.1 Descrição detalhada

Implementação do filtro FIR.

Autor

Arthur Damasceno

Atenção

Este código apresenta a implementação de filtro FIR com coeficientes calculados previamente

Definição no arquivo FIRFilter.c.

#### 4.1.2 Funções

#### 4.1.2.1 FIRFilter\_Init()

```
void FIRFilter_Init (
               FIRFilter * fir )
Definição na linha 19 do arquivo FIRFilter.c.
00019
00020
00021
           /* Reinicia o buffer do filtro */
00022
           for(uint8_t n=0; n<FIR_FILTER_LENGTH; n++) {</pre>
00023
               fir->buf[n] = 0.0f;
00024
00025
00026
          }
00027
00028
           /* Reseta o index do buffer */
00029
          fir->bufindex = 0;
00030
          /* Reseta a saída do filtro */
fir->out = 0.0f;
00031
00032
00033 }
```

#### 4.1.2.2 FIRFilter\_Update()

Definição na linha 35 do arquivo FIRFilter.c.

```
00035
00036
           /* Guarda a entrada no buffer */
00037
00038
           fir->buf[fir->bufindex] = inp;
00039
00040
           /\star Incrementa o index do buffer e reinicia se necessário \star/
00041
           fir->bufindex++;
00042
00043
           if(fir->bufindex == FIR_FILTER_LENGTH) {
00044
00045
               fir->bufindex = 0;
00046
00047
00048
00049
          /* Calcula nova saída via convolução */ fir->out = 0.0f;
00050
00051
00052
           uint8_t sumIndex = fir->bufindex;
00053
00054
           for (uint8_t n=0; n<FIR_FILTER_LENGTH; n++) {</pre>
00055
00056
               /* Decrementa o index e reseta se necessary */
00057
               if(sumIndex>0){
00058
00059
                   sumIndex--;
00060
00061
               }else{
00062
00063
                   sumIndex = FIR_FILTER_LENGTH -1;
00064
00065
00066
00067
               /\star Multiplica resposta ao impulso com a entrada deslocada e soma a saída \star/
00068
               fir->out += FIR_IMPULSE_RESPONSE[n] * fir->buf[sumIndex];
00069
           }
00070
00071
           /* Returna saída filtrada */
00072
           return fir->out;
00073
00074 }
```

4.2 FIRFilter.c 13

#### 4.2 FIRFilter.c

```
Vá para a documentação desse arquivo.
```

```
00014 #include "FIRFilter.h"
00016 /* Coeficientes do filtro */
00017 static float FIR_IMPULSE_RESPONSE[FIR_FILTER_LENGTH] = {0.02840647f, 0.23700821f, 0.46917063f,
       0.23700821f, 0.02840647f};
00018
00019 void FIRFilter_Init(FIRFilter *fir){
00020
00021
           /* Reinicia o buffer do filtro */
00022
           for (uint8_t n=0; n<FIR_FILTER_LENGTH; n++) {</pre>
00023
00024
               fir->buf[n] = 0.0f;
00025
00026
          }
00027
00028
           /* Reseta o index do buffer */
00029
          fir->bufindex = 0;
00030
00031
           /* Reseta a saída do filtro */
00032
          fir->out = 0.0f;
00033 }
00034
00035 float FIRFilter_Update(FIRFilter *fir, float inp){
00036
00037
           /* Guarda a entrada no buffer */
00038
          fir->buf[fir->bufindex] = inp;
00039
00040
           /\star Incrementa o index do buffer e reinicia se necessário \star/
00041
           fir->bufindex++;
00042
00043
          if(fir->bufindex == FIR FILTER LENGTH) {
00044
00045
               fir->bufindex = 0;
00046
00047
00048
00049
           /* Calcula nova saída via convolução */
00050
          fir->out = 0.0f;
00051
00052
          uint8_t sumIndex = fir->bufindex;
00053
00054
          for (uint8 t n=0; n<FIR FILTER LENGTH; n++) {</pre>
00055
00056
               /* Decrementa o index e reseta se necessary */
00057
               if(sumIndex>0){
00058
00059
                   sumIndex--;
00060
00061
               }else{
00062
00063
                   sumIndex = FIR FILTER LENGTH -1;
00064
00065
00066
               /* Multiplica resposta ao impulso com a entrada deslocada e soma a saída */ fir->out += FIR_IMPULSE_RESPONSE[n] * fir->buf[sumIndex];
00067
00068
00069
          }
00070
00071
           /* Returna saída filtrada */
00072
           return fir->out;
00073
00074 }
```

#### 4.3 freertos.c

```
00031 /* Private typedef ----
00032 /* USER CODE BEGIN PTD */
00033
00034 /* USER CODE END PTD */
00035
00036 /* Private define -----
00037 /* USER CODE BEGIN PD */
00038
00039 /* USER CODE END PD */
00040
00041 /* Private macro --
00042 /* USER CODE BEGIN PM */
00043
00044 /* USER CODE END PM */
00045
00046 /* Private variables -----
00047 /* USER CODE BEGIN Variables */
00049 /* USER CODE END Variables */
00050
00051 /* Private function prototypes -----
00052 /* USER CODE BEGIN FunctionPrototypes */
00053
00054 /* USER CODE END FunctionPrototypes */
00056 /* GetIdleTaskMemory prototype (linked to static allocation support) */
**ppxIdleTaskStackBuffer, uint32_t *pulIdleTaskStackSize);
00058
00059 /* USER CODE BEGIN GET_IDLE_TASK_MEMORY */
00060 static StaticTask_t xIdleTaskTCBBuffer;
00061 static StackType_t xIdleStack[configMINIMAL_STACK_SIZE];
00062
\texttt{00063 void vApplicationGetIdleTaskMemory(StaticTask\_t **ppxIdleTaskTCBBuffer, StackType\_terming the terminal transfer of the terminal transfer
               **ppxIdleTaskStackBuffer, uint32_t *pulIdleTaskStackSize )
00064 {
00065 *ppxIdleTaskTCBBuffer = &xIdleTaskTCBBuffer;
                *ppxIdleTaskStackBuffer = &xIdleStack[0];
00067
                 *pulIdleTaskStackSize = configMINIMAL_STACK_SIZE;
00068
               /* place for user code */
00069 }
00070 /* USER CODE END GET_IDLE_TASK_MEMORY */
00071
00072 /* Private application code -----
00073 /* USER CODE BEGIN Application */
00074
00075 /* USER CODE END Application */
00076
```

# 4.4 Referência do Arquivo D:/BACKUP/Faculdade/16\_Embarcados/ Controlador\_Temperatura/Controle-de-temperatura/ Codigos/controle temperatura RTOS/Core/Src/main.c

#### Corpo principal do programa.

```
#include "main.h"
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "string.h"
#include "stdio.h"
#include "stdlib.h"
#include "FIRFilter.h"
```

#### Definições e Macros

#define r 0.000210115034038755f

15

- #define p 1.0f
- #define k 0.0638221651196478f
- #define ksat 0.1f
- #define CSen HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_15, GPIO\_PIN\_RESET);
- #define CSdis HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_15, GPIO\_PIN\_SET);
- #define SCK H HAL GPIO WritePin(GPIOB, GPIO PIN 3, GPIO PIN RESET);
- #define SCK L HAL GPIO WritePin(GPIOB, GPIO PIN 3, GPIO PIN SET);

#### **Funções**

void SystemClock Config (void)

Configuração do clock do sistema.

void Temp\_taskF (void \*pvParameters)

Tarefa de leitura da temperatura.

void Filter\_taskF (void \*pvParameters)

Tarefa de filtro da variável temperatura.

void Control\_taskF (void \*pvParameters)

Tarefa de atualização da referencia, calculo e execução da lei de controle.

void Display\_taskF (void \*pvParameters)

Tarefa de atualização das variáveis no display.

• int main (void)

ponto de entrada da aplicação.

void HAL\_TIM\_PeriodElapsedCallback (TIM\_HandleTypeDef \*htim)

chamada da função de período

void Error Handler (void)

Função executada em caso de erro na aplicação.

#### **Variáveis**

- ADC HandleTypeDef hadc1
- TIM HandleTypeDef htim1
- UART HandleTypeDef huart1
- · TaskHandle t Temp Task
- TaskHandle\_t Filter\_Task
- TaskHandle t Control Task
- · TaskHandle t Display Task
- QueueHandle\_t tempQueue
- QueueHandle t filteredTempQueue
- FIRFilter tempFilter
- float filteredTemp = 0
- float ref = 0
- float dutyCycle = 0

#### 4.4.1 Descrição detalhada

Corpo principal do programa.

**Autor** 

Arthur Damasceno

Mateus Piccinin

Atenção

#### Controlador de temperatura

Este código apresenta a implementação de um controlador de temperatura utilizando a interface FreeRTOS.

Definição no arquivo main.c.

#### 4.4.2 Definições e macros

#### 4.4.2.1 CSdis

```
#define CSdis HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, GPIO_PIN_SET);
```

Definição na linha 36 do arquivo main.c.

#### 4.4.2.2 CSen

```
#define CSen HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, GPIO_PIN_RESET);
```

Definição na linha 35 do arquivo main.c.

#### 4.4.2.3 k

#define k 0.0638221651196478f

Definição na linha 32 do arquivo main.c.

#### 4.4.2.4 ksat

#define ksat 0.1f

Definição na linha 33 do arquivo main.c.

#### 4.4.2.5 p

```
#define p 1.0f
```

Definição na linha 31 do arquivo main.c.

#### 4.4.2.6 r

```
#define r 0.000210115034038755f
```

Definição na linha 30 do arquivo main.c.

#### 4.4.2.7 SCK\_H

```
#define SCK_H HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, GPIO_PIN_RESET);
```

Definição na linha 38 do arquivo main.c.

#### 4.4.2.8 SCK\_L

```
#define SCK_L HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, GPIO_PIN_SET);
```

Definição na linha 39 do arquivo main.c.

#### 4.4.3 Funções

#### 4.4.3.1 Control\_taskF()

Tarefa de atualização da referencia, calculo e execução da lei de controle.

#### Observação

Esta tarefa executa o calculo da lei de controle, atualizando o valor de referencia setando a razão cíclica da chave de saída ou ativando a ventoinha de resfrianmento

#### **Parâmetros**

\*pvParameters (não utilizado) permite iniciar a função com valor inical

#### Valores Retornados

None

```
Definição na linha 386 do arquivo main.c.
00387
          while (1) {
00388
               float rx_filteredTemp;
               /* Recebe da fila filteredTempQueue */
00389
00390
               if (xQueueReceive(filteredTempQueue, &rx_filteredTemp, 10)) {
                   /* Leitura da entrada analógica para calculo de referencia */
00391
00392
                   HAL_ADC_PollForConversion(&hadc1, 10);
                   ref = (float) HAL_ADC_GetValue(&hadc1) / 27.3; // leitura do potenciometro convertido em
00393
       ref até 150°C
00394
00395
                   /* Lei de controle */
00396
                   float u;
00397
                   static float up, uint;
                   int flag_sat;
00398
00399
                   float ek = ref - rx_filteredTemp;
00400
00401
                   /\star Controlador bang-bang ventoinha \star/
00402
                   if (ek < -15.0) {
00403
                       HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, GPIO_PIN_SET);
00404
00405
                       HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, GPIO_PIN_RESET);
                   }
00406
00407
00408
                   /* Anti-windup integrador */
                   if (!flag_sat) {
00409
00410
                       uint = uint * p + r * ek;
00411
00412
                   } else if (flag_sat) {
                       uint = (uint * p + r * ek) * ksat;
00413
                   }
00414
00415
00416
                   /* Proporcional */
00417
00418
00419
                   /* Ação de controle */
00420
                   u = up + uint;
00421
00422
                   /* Conversão período PWM */
00423
                   u = u * 4500.0;
00424
                   /* Limites de saturação de PWM */ if (u > 18000.0) {
00425
00426
00427
                       u = 18000.0;
                   flag_sat = 1;
} else if (u < 0) {
u = 0;
00429
00430
00431
                       flag_sat = 1;
00432
                   } else {
    u = u;
00433
00434
                       flag_sat = 0;
00435
00436
                   /* Converte periodo do timer em razão cíclica */ dutyCycle = u / 180.0;
00437
00438
00439
00440
                   /* Seta periférico PWM */
00441
                   __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, (uint32_t ) u);
00442
00443
               }
00444
00445
          }
```

#### 4.4.3.2 Display\_taskF()

00446 }

```
void Display_taskF (
```

```
void * pvParameters )
```

Tarefa de atualização das variáveis no display.

#### Observação

Esta tarefa envia ao display TFT via UART os valores atualizados de referência, variável manipulada(razão cíclica) e variável de processo (temperatura)

#### **Parâmetros**

\*pvParameters (não utilizado) permite iniciar a função com valor inical

#### **Valores Retornados**

Definição na linha 456 do arquivo main.c.

```
00456
00457
            while (1) {
           char str[100];
00458
                 /* Fim de comando definido pela API do display */
00459
                 uint8_t Cmd_End[3] = { 0xFF, 0xFF, 0xFF };
00460
00461
00462
                  /* Atualiza valor do setpoint */
00463
                 int32_t number = ref \star 100;
00464
                 sprintf(str, "setPoint.val=%ld", number);
00465
                 HAL_UART_Transmit(&huart1, (uint8_t*) str, strlen(str), 10);
HAL_UART_Transmit(&huart1, Cmd_End, 3, 10);
00466
00467
00468
                  /\star Atualiza valor da variável de processo \star/
                 number = filteredTemp * 100;
sprintf(str, "filteredTemp.val=%ld", number);
00469
00470
                 HAL_UART_Transmit(&huart1, (uint8_t*) str, strlen(str), 10);
HAL_UART_Transmit(&huart1, Cmd_End, 3, 10);
00471
00472
00474
                  /∗ Atualiza valor da variável manipulada ∗/
                 number = dutyCycle * 100;
sprintf(str, "dutyCycle.val=%ld", number);
HAL_UART_Transmit(&huart1, (uint8_t*) str, strlen(str), 10);
HAL_UART_Transmit(&huart1, Cmd_End, 3, 10);
00475
00476
00477
00478
00479
00480
                  /* Atraso para definição do período da tarefa */
00481
                  vTaskDelay(1000); /*1Hz frequency*/
00482
            }
00483 }
```

#### 4.4.3.3 Error\_Handler()

Função executada em caso de erro na aplicação.

#### Valores Retornados

None

Definição na linha 501 do arquivo main.c.

#### 4.4.3.4 Filter\_taskF()

Tarefa de filtro da variável temperatura.

#### Observação

Esta tarefa executa a chamada para o filtro FIR, após 5 atualizações o valor é adicionado a fila filteredTemp⇔ Queue

#### **Parâmetros**

\*pvParameters (não utilizado) permite iniciar a função com valor inical

#### Valores Retornados

None

Definição na linha 357 do arquivo main.c.

```
00358
           uint8_t aux = 0;
00359
           while (1) {
               float rx_temp;
/* Recebe da fila filteredTempQueue */
00360
00361
00362
                if (xQueueReceive(tempQueue, &rx_temp, 10)) {
00363
00364
                     /* Chamada do filtro FIR */
00365
                     filteredTemp = FIRFilter_Update(&tempFilter, rx_temp);
00366
00367
00368
                if (aux == 5) {
00369
00370
                    aux = 0;
                    /* Addiciona a fila filteredTempQueue */
if (xQueueSend(filteredTempQueue, &filteredTemp, 10) == pdPASS) {
00371
00372
00373
00374
00375
           }
00376 }
```

#### 4.4.3.5 HAL\_TIM\_PeriodElapsedCallback()

```
void HAL_TIM_PeriodElapsedCallback ( {\tt TIM\_HandleTypeDef} \ * \ htim \ )
```

chamada da função de período

#### Observação

Esta função atualiza o valor de "uwTick" utilizado como base de tempo do sistema

#### **Parâmetros**

htim	: TIM handle
------	--------------

#### Valores Retornados

```
None
```

#### Definição na linha 491 do arquivo main.c.

#### 4.4.3.6 main()

```
int main (
     void )
```

ponto de entrada da aplicação.

#### Valores Retornados



#### Definição na linha 76 do arquivo main.c.

```
00076
00077
           /\star Reinicia todos os periféricos, inicializa a interface flash e o systick \star/
00078
          HAL_Init();
00079
08000
           /* Configura o clock do sistema */
00081
           SystemClock_Config();
00082
00083
           /\star Inicializa todos os periféricos configurados \star/
          MX_GPIO_Init();
MX_ADC1_Init();
MX_TIM1_Init();
00084
00085
00086
00087
          MX_USART1_UART_Init();
88000
00089
           /* Inicializa o conversor AD */
          if (HAL_ADC_Start(&hadc1) != HAL_OK) {
00090
00091
              Error_Handler();
00092
00093
00094
          /\star Inicializa o timer1 em modo PWM \star/
00095
          if (HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1) != HAL_OK) {
00096
               Error_Handler();
00097
00098
00099
           /* Inicializa o filtro FIR */
00100
          FIRFilter_Init(&tempFilter);
00101
           /\star Cria a fila de leituras de temperatura bruta \star/
00102
          tempQueue = xQueueCreate(1, sizeof(float));
if (tempQueue == 0) {
00103
00104
00105
               Error_Handler();
00106
00107
           /\star Cria a fila de leituras de temperatura filtrada \star/
00108
00109
          filteredTempQueue = xQueueCreate(1, sizeof(float));
          if (tempQueue == 0) {
00110
00111
               Error_Handler();
00112
00113
```

```
/* Cria tasks na pilha do sistema */
                   /* Cital tasks ha pilla do Sistema */
xTaskCreate(Temp_taskF, "TempTask", 128, NULL, 3, &Temp_Task);
xTaskCreate(Filter_taskF, "FilterTask", 128, NULL, 2, &Filter_Task);
xTaskCreate(Control_taskF, "ControlTask", 128, NULL, 4, &Control_Task);
xTaskCreate(Display_taskF, "DisplayTask", 128, NULL, 1, &Display_Task);
00115
00116
00117
00118
00119
00120
                     /* Inicializa o escalonador */
00121
                    vTaskStartScheduler();
00122
                   /* Loop infinito */
while (1) {
00123
00124
00125
00126 }
```

#### 4.4.3.7 SystemClock\_Config()

Configuração do clock do sistema.

Valores Retornados

```
None
```

Definição na linha 132 do arquivo main.c.

```
00133
          RCC_OscInitTypeDef RCC_OscInitStruct = { 0 };
00134
          RCC_ClkInitTypeDef RCC_ClkInitStruct = { 0 };
00135
          RCC_PeriphCLKInitTypeDef PeriphClkInit = { 0 };
00136
          RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
00137
          RCC_OscInitStruct.HSEState = RCC_HSE_ON;
RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
00138
00139
00140
          RCC_OscInitStruct.HSIState = RCC_HSI_ON;
00141
          RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
          RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
00142
00143
00144
          if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK) {
00145
              Error Handler();
00146
00147
          00148
00149
00150
00151
00152
          RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
00153
          RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
00154
          if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK) {
00155
00156
              Error_Handler();
00157
00158
          PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_ADC;
00159
          PeriphClkInit.AdcClockSelection = RCC_ADCPCLK2_DIV6;
00160
          if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK) {
00161
              Error_Handler();
00162
00163 }
```

#### 4.4.3.8 Temp\_taskF()

Tarefa de leitura da temperatura.

Observação

Esta tarefa executa a leitura da temperatura armazenada na memória do módulo MAX6675 atravez de um bitbanging do protocolo SPI, ao fim da conversão o valor é adicionado a fila tempQueue

#### **Parâmetros**

\*pvParameters (não utilizado) permite iniciar a função com valor inical

#### **Valores Retornados**

None

Definição na linha 315 do arquivo main.c.

```
00315
00316
00317
              uint8_t tempdata[16];
00318
              uint16_t temp16 = 0;
00319
00320
               /* bitbanging protocolo SPI */
00321
              CSen
              for (int i = 0; i < 16; i++) {</pre>
00322
00323
00324
                   tempdata[i] = HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_4);
00325
                   SCK_L
00326
00327
              CSdis
00328
00329
              /* Conversão temperatura */
00330
              if (tempdata[13] == 0) {
00331
                  for (int n = 1; n < 13; n++) { temp16 += tempdata[n] * (2048 / (1 * (n - 1)));
00332
00333
00334
00335
00336
              }
00337
              float temp = (float) temp16 / 4;
00338
00339
              /* Adiciona a fila tempQueue */
00340
00341
               if (xQueueSend(tempQueue, &temp, 10) == pdPASS) {
00342
00343
00344
              /\star Atraso para definição do período da tarefa \star/
00345
              vTaskDelay(200); /*5Hz frequency*/
00346
          }
00347 }
```

#### 4.4.4 Variáveis

#### 4.4.4.1 Control\_Task

TaskHandle\_t Control\_Task

Definição na linha 49 do arquivo main.c.

#### 4.4.4.2 Display\_Task

```
{\tt TaskHandle\_t\ Display\_Task}
```

Definição na linha 50 do arquivo main.c.

#### 4.4.4.3 dutyCycle

```
float dutyCycle = 0
```

Definição na linha 59 do arquivo main.c.

#### 4.4.4.4 Filter\_Task

```
TaskHandle_t Filter_Task
```

Definição na linha 48 do arquivo main.c.

#### 4.4.4.5 filteredTemp

```
float filteredTemp = 0
```

Definição na linha 57 do arquivo main.c.

#### 4.4.4.6 filteredTempQueue

```
QueueHandle_t filteredTempQueue
```

Definição na linha 53 do arquivo main.c.

#### 4.4.4.7 hadc1

ADC\_HandleTypeDef hadc1

Definição na linha 41 do arquivo main.c.

TIM\_HandleTypeDef htim1

Definição na linha 43 do arquivo main.c.

#### 4.4.4.9 huart1

UART\_HandleTypeDef huart1

Definição na linha 45 do arquivo main.c.

#### 4.4.4.10 ref

float ref = 0

Definição na linha 58 do arquivo main.c.

#### 4.4.4.11 Temp\_Task

TaskHandle\_t Temp\_Task

Definição na linha 47 do arquivo main.c.

#### 4.4.4.12 tempFilter

FIRFilter tempFilter

Definição na linha 55 do arquivo main.c.

#### 4.4.4.13 tempQueue

QueueHandle\_t tempQueue

Definição na linha 52 do arquivo main.c.

#### 4.5 main.c

Vá para a documentação desse arquivo.

```
00001
00018 #include "main.h"
00019
00020 #include "FreeRTOS.h"
00021 #include "task.h"
00022 #include "queue.h"
00023
00024 #include "string.h"
00025 #include "stdio.h"
00026 #include "stdlib.h"
00027
00028 #include "FIRFilter.h"
00029
00030 #define r 0.000210115034038755f
00031 #define p 1.0f
00032 #define k 0.0638221651196478f
00033 #define ksat 0.1f
00034
00035 #define CSen HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, GPIO_PIN_RESET);
00036 #define CSdis HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, GPIO_PIN_SET);
00037
00038 #define SCK_H HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, GPIO_PIN_RESET);
00039 #define SCK_L HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, GPIO_PIN_SET);
00040
00041 ADC_HandleTypeDef hadc1;
00042
00043 TIM_HandleTypeDef htim1;
00044
00045 UART_HandleTypeDef huart1;
00046
00047 TaskHandle_t Temp_Task;
00048 TaskHandle_t Filter_Task;
00049 TaskHandle_t Control_Task;
00050 TaskHandle_t Display_Task;
00051
00052 QueueHandle_t tempQueue;
00053 QueueHandle_t filteredTempQueue;
00054
00055 FIRFilter tempFilter;
00056
00057 float filteredTemp = 0;
00058 float ref = 0;
00059 float dutyCycle = 0;
00060
00061 void SystemClock Config(void);
00062 static void MX_GPIO_Init(void);
00063 static void MX_ADC1_Init(void);
00064 static void MX_TIM1_Init(void);
00065 static void MX_USART1_UART_Init(void);
00066
00067 void Temp_taskF(void *pvParameters);
00068 void Filter_taskF(void *pvParameters);
00069 void Control_taskF(void *pvParameters);
00070 void Display_taskF(void *pvParameters);
00071
00076 int main(void) {
00077
           /\star Reinicia todos os periféricos, inicializa a interface flash e o systick \star/
00078
          HAL Init();
00079
           /* Configura o clock do sistema */
00081
          SystemClock_Config();
00082
00083
           /* Inicializa todos os periféricos configurados */
00084
          MX\_GPIO\_Init();
          MX_ADC1_Init();
00085
          MX_TIM1_Init();
00086
00087
          MX_USART1_UART_Init();
00088
00089
           /* Inicializa o conversor AD */
          if (HAL_ADC_Start(&hadc1) != HAL_OK) {
00090
00091
               Error_Handler();
00092
00093
00094
           /* Inicializa o timer1 em modo PWM */
00095
          if (HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1) != HAL_OK) {
00096
               Error_Handler();
00097
00098
00099
           /* Inicializa o filtro FIR */
00100
          FIRFilter_Init(&tempFilter);
00101
00102
          /* Cria a fila de leituras de temperatura bruta */
```

4.5 main.c 27

```
tempQueue = xQueueCreate(1, sizeof(float));
           if (tempQueue == 0) {
00104
00105
               Error_Handler();
00106
00107
00108
           /* Cria a fila de leituras de temperatura filtrada */
           filteredTempQueue = xQueueCreate(1, sizeof(float));
00109
00110
           if (tempQueue == 0) {
00111
               Error_Handler();
00112
00113
00114
           /\star Cria tasks na pilha do sistema \star/
           xTaskCreate(Temp_taskF, "TempTask", 128, NULL, 3, &Temp_Task);
xTaskCreate(Filter_taskF, "FilterTask", 128, NULL, 2, &Filter_Task);
xTaskCreate(Control_taskF, "ControlTask", 128, NULL, 4, &Control_Task);
00115
00116
00117
           xTaskCreate(Display_taskF, "DisplayTask", 128, NULL, 1, &Display_Task);
00118
00119
00120
           /* Inicializa o escalonador */
           vTaskStartScheduler();
00121
00122
00123
           /* Loop infinito */
00124
           while (1) {
00125
           }
00126 }
00127
00132 void SystemClock_Config(void) {
00133
           RCC_OscInitTypeDef RCC_OscInitStruct = { 0 };
00134
           RCC_ClkInitTypeDef RCC_ClkInitStruct = { 0 };
00135
           RCC_PeriphCLKInitTypeDef PeriphClkInit = { 0 };
00136
           RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
RCC_OscInitStruct.HSEState = RCC_HSE_ON;
00137
00138
00139
           RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
00140
           RCC_OscInitStruct.HSIState = RCC_HSI_ON;
           RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
00141
00142
           RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
00143
           if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK) {
00145
               Error_Handler();
00146
00147
00148
           RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLK
                    | RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2;
00149
           RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
00150
           RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
00151
00152
           RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
00153
           RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
00154
00155
           if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK) {
00156
               Error Handler():
00157
00158
           PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_ADC;
00159
           PeriphClkInit.AdcClockSelection = RCC_ADCPCLK2_DIV6;
00160
           if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK) {
00161
               Error_Handler();
00162
00163 }
00164
00170 static void MX_ADC1_Init(void) {
00171
00172
           ADC ChannelConfTypeDef sConfig = { 0 };
00173
00174
           hadc1.Instance = ADC1;
00175
           hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
00176
           hadc1.Init.ContinuousConvMode = ENABLE;
           hadc1.Init.DiscontinuousConvMode = DISABLE;
hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
00177
00178
           hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
00179
00180
           hadc1.Init.NbrOfConversion = 1;
00181
           if (HAL_ADC_Init(&hadc1) != HAL_OK) {
00182
               Error_Handler();
00183
00184
           sConfig.Channel = ADC_CHANNEL_0;
00185
           sConfig.Rank = ADC_REGULAR_RANK_1;
sConfig.SamplingTime = ADC_SAMPLETIME_28CYCLES_5;
00186
00187
00188
           if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK) {
00189
               Error_Handler();
00190
00191 }
00192
00198 static void MX_TIM1_Init(void) {
00199
00200
           TIM_MasterConfigTypeDef sMasterConfig = { 0 };
00201
           TIM_OC_InitTypeDef sConfigOC = { 0 };
           TIM_BreakDeadTimeConfigTypeDef sBreakDeadTimeConfig = { 0 };
00202
00203
```

```
htim1.Instance = TIM1;
00204
00205
           htim1.Init.Prescaler = 1;
           htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
00206
          htim1.Init.Period = 18000 - 1;
htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
00207
00208
          htim1.Init.RepetitionCounter = 0;
htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
00209
00210
00211
           if (HAL_TIM_PWM_Init(&htim1) != HAL_OK) {
00212
               Error_Handler();
00213
00214
           sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
           sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
00215
00216
           if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig)
00217
                    != HAL_OK) {
00218
               Error_Handler();
00219
           sConfigOC.OCMode = TIM_OCMODE_PWM1;
00220
           sConfigOC.Pulse = 0;
00221
           sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
           sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
00223
00224
           sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
           sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
00225
00226
           if (HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC, TIM_CHANNEL_1)
00227
00228
                    != HAL_OK) {
               Error_Handler();
00229
00230
00231
           sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
           sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
00232
           sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF; sBreakDeadTimeConfig.DeadTime = 0;
00233
00234
00235
           sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
00236
           sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
           sBreakDeadTimeConfig.AutomaticOutput = TIM_AUTOMATICOUTPUT_DISABLE;
00237
00238
           if (HAL_TIMEx_ConfigBreakDeadTime(&htim1, &sBreakDeadTimeConfig)
00239
                    ! = HAL OK) {
00240
               Error Handler();
00241
00242
00243
           HAL_TIM_MspPostInit(&htim1);
00244 }
00245
00251 static void MX USART1 UART Init (void) {
00252
00253
           huart1.Instance = USART1;
00254
           huart1.Init.BaudRate = 115200;
00255
           huart1.Init.WordLength = UART_WORDLENGTH_8B;
          huart1.Init.StopBits = UART_STOPBITS_1;
huart1.Init.Parity = UART_PARITY_NONE;
huart1.Init.Mode = UART_MODE_TX;
00256
00257
00258
           huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
00260
           huart1.Init.OverSampling = UART_OVERSAMPLING_16;
00261
           if (HAL_UART_Init(&huart1) != HAL_OK) {
00262
               Error_Handler();
00263
00264 }
00271 static void MX_GPIO_Init(void) {
00272
           GPIO_InitTypeDef GPIO_InitStruct = { 0 };
00273
00274
            _HAL_RCC_GPIOC_CLK_ENABLE();
           __HAL_RCC_GPIOD_CLK_ENABLE();
00275
00276
           __HAL_RCC_GPIOA_CLK_ENABLE();
00277
           __HAL_RCC_GPIOB_CLK_ENABLE();
00278
00279
           HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, GPIO_PIN_RESET);
00280
00281
           HAL GPIO WritePin (GPIOB, GPIO PIN 3, GPIO PIN RESET);
00282
00283
           GPIO_InitStruct.Pin = GPIO_PIN_15;
           GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
00284
00285
00286
           GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
00287
           HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
00288
00289
           GPIO_InitStruct.Pin = GPIO_PIN_3;
00290
           GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
00291
           GPIO_InitStruct.Pull = GPIO_NOPULL;
           GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
00292
           HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
00293
00294
00295
           GPIO_InitStruct.Pin = GPIO_PIN_4;
           GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
00296
00297
00298
           HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
00299
00300
           GPIO_InitStruct.Pin = GPIO_PIN_15;
```

4.5 main.c 29

```
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
00302
          GPIO_InitStruct.Pull = GPIO_NOPULL;
          GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
00303
00304
          HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
00305 }
00306
00315 void Temp_taskF(void *pvParameters) {
00316
          while (1) {
00317
              uint8_t tempdata[16];
00318
               uint16_t temp16 = 0;
00319
               /* bitbanging protocolo SPI */
00320
00321
               CSen
00322
               for (int i = 0; i < 16; i++) {</pre>
00323
                   SCK_H
                   tempdata[i] = HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_4);
00324
00325
                   SCK_L
00326
00327
               CSdis
00328
00329
               /* Conversão temperatura */
00330
               if (tempdata[13] == 0) {
00331
                   for (int n = 1; n < 13; n++) {
    temp16 += tempdata[n] * (2048 / (1 \ll (n - 1)));
00332
00333
00334
00335
00336
00337
00338
               float temp = (float) temp16 / 4;
00339
00340
               /* Adiciona a fila tempQueue */
00341
               if (xQueueSend(tempQueue, &temp, 10) == pdPASS) {
00342
00343
               /★ Atraso para definição do período da tarefa */
00344
00345
              vTaskDelay(200); /*5Hz frequency*/
00346
00347 }
00348
00357 void Filter_taskF(void *pvParameters) {
00358
          uint8_t aux = 0;
          while (1) {
00359
00360
               float rx_temp;
00361
               /* Recebe da fila filteredTempQueue */
00362
               if (xQueueReceive(tempQueue, &rx_temp, 10)) {
00363
                   aux++;
                   /* Chamada do filtro FIR */
00364
                   filteredTemp = FIRFilter_Update(&tempFilter, rx_temp);
00365
00366
00367
00368
               if (aux == 5) {
00369
00370
                   aux = 0;
00371
                   /* Adiciona a fila filteredTempQueue */
00372
                   if (xQueueSend(filteredTempQueue, &filteredTemp, 10) == pdPASS) {
00374
               }
00375
          }
00376 }
00377
00386 void Control_taskF(void *pvParameters) {
00387
          while (1) {
00388
              float rx_filteredTemp;
00389
               /* Recebe da fila filteredTempQueue */
00390
               if (xQueueReceive(filteredTempQueue, &rx_filteredTemp, 10)) {
00391
                   /\star Leitura da entrada analógica para calculo de referencia \star/
                   HAL_ADC_PollForConversion(&hadcl, 10);
00392
                   ref = (float) HAL_ADC_GetValue(&hadc1) / 27.3; // leitura do potenciometro convertido em
00393
       ref até 150°C
00394
00395
                   /\star Lei de controle \star/
00396
                   float u;
                   static float up, uint;
00397
                   int flag_sat;
float ek = ref - rx_filteredTemp;
00398
00399
00400
00401
                   /* Controlador bang-bang ventoinha */
00402
                   if (ek < -15.0) {
                       HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, GPIO_PIN_SET);
00403
00404
                   } else {
00405
                       HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, GPIO_PIN_RESET);
00406
00407
00408
                   /* Anti-windup integrador */
00409
                   if (!flag_sat) {
00410
                       uint = uint * p + r * ek;
```

```
} else if (flag_sat) {
    uint = (uint * p + r * ek) * ksat;
00412
00413
00414
00415
00416
                     /* Proporcional */
                    up = k * ek;
00418
00419
                     /* Ação de controle */
00420
                    u = up + uint;
00421
00422
                     /* Conversão período PWM */
00423
                    u = u * 4500.0;
00424
00425
                     /\star Limites de saturação de PWM \star/
                    if (u > 18000.0) {
u = 18000.0;
00426
00427
00428
                         flag_sat = 1;
                     } else if (u < 0) {
u = 0;
00430
00431
                         flag_sat = 1;
00432
                     } else {
                         11 = 11:
00433
                         flag_sat = 0;
00434
00435
00436
00437
                     /\star Converte periodo do timer em razão cíclica \star/
00438
                    dutyCycle = u / 180.0;
00439
                    /* Seta periférico PWM */
00440
                     __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, (uint32_t ) u);
00441
00442
00443
00444
00445
           }
00446 }
00447
00456 void Display_taskF(void *pvParameters) {
00457
          while (1) {
00458
             char str[100];
                /\star Fim de comando definido pela API do display \star/
00459
00460
                uint8_t Cmd_End[3] = { 0xFF, 0xFF, 0xFF };
00461
00462
                /* Atualiza valor do setpoint */
00463
                int32_t number = ref * 100;
00464
                sprintf(str, "setPoint.val=%ld", number);
                HAL_UART_Transmit(&huart1, (uint8_t*) str, strlen(str), 10);
HAL_UART_Transmit(&huart1, Cmd_End, 3, 10);
00465
00466
00467
00468
                /* Atualiza valor da variável de processo */
00469
                number = filteredTemp * 100;
00470
                sprintf(str, "filteredTemp.val=%ld", number);
                HAL_UART_Transmit(&huart1, (uint8_t*) str, strlen(str), 10);
HAL_UART_Transmit(&huart1, Cmd_End, 3, 10);
00471
00472
00473
00474
                /* Atualiza valor da variável manipulada */
                number = dutyCycle * 100;
sprintf(str, "dutyCycle.val=%ld", number);
00475
00476
                HAL_UART_Transmit(&huart1, (uint8_t*) str, strlen(str), 10);
HAL_UART_Transmit(&huart1, Cmd_End, 3, 10);
00477
00478
00479
                /* Atraso para definição do período da tarefa */ vTaskDelay(1000); /*1Hz frequency*/
00480
00481
00482
           }
00483 }
00484
00491 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
          if (htim->Instance == TIM4) {
00492
00493
               HAL_IncTick();
00494
           }
00495 }
00496
00501 void Error_Handler(void) {
          __disable_irq();
while (1) {
00502
00503
00504
00505 }
00506
00507 #ifdef USE_FULL_ASSERT
00514 void assert_failed(uint8_t *file, uint32_t line)
00515 {
00517 #endif
```

# 4.6 Referência do Arquivo

D:/BACKUP/Faculdade/16\_Embarcados/Controlador\_Temperatura/
Controle-de-temperatura/Codigos/controle\_temperatura\_RTOS/
Core/Src/stm32f1xx hal msp.c

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

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

# **Funções**

- void HAL\_TIM\_MspPostInit (TIM\_HandleTypeDef \*htim)
- void HAL MspInit (void)
- void HAL ADC MspInit (ADC HandleTypeDef \*hadc)

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

void HAL\_ADC\_MspDeInit (ADC\_HandleTypeDef \*hadc)

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

void HAL\_TIM\_PWM\_MspInit (TIM\_HandleTypeDef \*htim\_oc)

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

void HAL\_TIM\_PWM\_MspDeInit (TIM\_HandleTypeDef \*htim\_oc)

TIM\_OC MSP De-Initialization This function freeze the hardware resources used in this example.

void HAL\_UART\_MspInit (UART\_HandleTypeDef \*huart)

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

void HAL UART MspDeInit (UART HandleTypeDef \*huart)

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

# 4.6.1 Descrição detalhada

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

Atenção

# © Copyright (c) 2021 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

Definição no arquivo stm32f1xx hal msp.c.

# 4.6.2 Funções

# 4.6.2.1 HAL\_ADC\_MspDeInit()

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

#### **Parâmetros**

hadc ADC handle pointer

#### Valores Retornados

None

ADC1 GPIO Configuration PA0-WKUP ----> ADC1 IN0

Definição na linha 126 do arquivo stm32f1xx\_hal\_msp.c.

```
00128
        if (hadc->Instance==ADC1)
00129
        /* USER CODE BEGIN ADC1_MspDeInit 0 */
00130
00131
00132
        /* USER CODE END ADC1_MspDeInit 0 */
00133
         /* Peripheral clock disable */
00134
          __HAL_RCC_ADC1_CLK_DISABLE();
00135
00139
          HAL_GPIO_DeInit(GPIOA, GPIO_PIN_0);
00140
00141
        /* USER CODE BEGIN ADC1_MspDeInit 1 */
00142
00143
        /\star USER CODE END ADC1_MspDeInit 1 \star/
00144
00145
00146 }
```

# 4.6.2.2 HAL\_ADC\_MspInit()

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

# **Parâmetros**

hadc ADC handle pointer

#### Valores Retornados

None

ADC1 GPIO Configuration PA0-WKUP ----> ADC1 IN0

Definição na linha 94 do arquivo stm32f1xx\_hal\_msp.c.

```
00095 {
00096
        GPIO_InitTypeDef GPIO_InitStruct = {0};
00097
        if (hadc->Instance==ADC1)
00098
00099
        /* USER CODE BEGIN ADC1_MspInit 0 */
00100
00101
        /* USER CODE END ADC1_MspInit 0 */
          /* Peripheral clock enable */
__HAL_RCC_ADC1_CLK_ENABLE();
00102
00103
00104
00105
            _HAL_RCC_GPIOA_CLK_ENABLE();
00109
           GPIO_InitStruct.Pin = GPIO_PIN_0;
```

# 4.6.2.3 HAL\_MspInit()

00118 }

```
void HAL_MspInit (
     void )
```

Initializes the Global MSP. DISABLE: JTAG-DP Disabled and SW-DP Disabled

Definição na linha 66 do arquivo stm32f1xx hal msp.c.

```
00067 {
        /* USER CODE BEGIN MspInit 0 */
00069
       /* USER CODE END MspInit 0 */
00070
00071
00072
         __HAL_RCC_AFIO_CLK_ENABLE();
00073
        __HAL_RCC_PWR_CLK_ENABLE();
00074
00075
       /* System interrupt init*/
00076
        /* PendSV_IRQn interrupt configuration */
00077
       HAL_NVIC_SetPriority(PendSV_IRQn, 15, 0);
00078
        __HAL_AFIO_REMAP_SWJ_DISABLE();
00081
00082
00083
       /* USER CODE BEGIN MspInit 1 */
00084
00085
       /* USER CODE END MspInit 1 */
00086 }
```

#### 4.6.2.4 HAL TIM MspPostInit()

TIM1 GPIO Configuration PA8 ----> TIM1\_CH1

Definição na linha 170 do arquivo stm32f1xx\_hal\_msp.c.

```
00171 {
00172
       GPIO_InitTypeDef GPIO_InitStruct = {0};
00173
       if (htim->Instance==TIM1)
00174
       /* USER CODE BEGIN TIM1_MspPostInit 0 */
00176
00177
       /* USER CODE END TIM1_MspPostInit 0 */
00178
00179
           _HAL_RCC_GPIOA_CLK_ENABLE();
        GPIO_InitStruct.Pin = GPIO_PIN_8;
00183
        GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00184
00185
         GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
00186
         HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
00187
00188
       /* USER CODE BEGIN TIM1_MspPostInit 1 */
00189
00190
       /* USER CODE END TIM1_MspPostInit 1 */
00191
00192
00193 }
```

#### 4.6.2.5 HAL\_TIM\_PWM\_MspDeInit()

TIM\_OC MSP De-Initialization This function freeze the hardware resources used in this example.

# **Parâmetros**

```
htim_oc TIM_OC handle pointer
```

#### **Valores Retornados**

```
None
```

Definição na linha 200 do arquivo stm32f1xx\_hal\_msp.c.

```
if (htim_oc->Instance==TIM1)
00202
00203
        /* USER CODE BEGIN TIM1_MspDeInit 0 */
00204
00205
00206
       /* USER CODE END TIM1_MspDeInit 0 */
00207
         /* Peripheral clock disable */
00208
          __HAL_RCC_TIM1_CLK_DISABLE();
00209
       /* USER CODE BEGIN TIM1_MspDeInit 1 */
00210
00211
        /* USER CODE END TIM1_MspDeInit 1 */
00212
00213
00214 }
```

#### 4.6.2.6 HAL\_TIM\_PWM\_MspInit()

TIM\_OC MSP Initialization This function configures the hardware resources used in this example.

# **Parâmetros**

htim_oc	TIM_OC handle pointer

# Valores Retornados

None

Definição na linha 154 do arquivo stm32f1xx\_hal\_msp.c.

```
00155 {
00156     if (htim_oc->Instance==TIM1)
00157     {
00158     /* USER CODE BEGIN TIM1_MspInit 0 */
00159
00160     /* USER CODE END TIM1_MspInit 0 */
00161     /* Peripheral clock enable */
00162     __HAL_RCC_TIM1_CLK_ENABLE();
00163     /* USER CODE BEGIN TIM1_MspInit 1 */
00164
```

# 4.6.2.7 HAL\_UART\_MspDeInit()

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

#### **Parâmetros**

huart	UART handle pointer
-------	---------------------

#### Valores Retornados

None

USART1 GPIO Configuration PA9 ----> USART1\_TX PA10 ----> USART1\_RX

Definição na linha 261 do arquivo stm32f1xx\_hal\_msp.c.

```
00262 {
00263
        if (huart->Instance==USART1)
00264
00265
        /* USER CODE BEGIN USART1_MspDeInit 0 */
00266
       /* USER CODE END USART1_MspDeInit 0 */
00267
        /* Peripheral clock disable */
__HAL_RCC_USART1_CLK_DISABLE();
00268
00269
00275
        HAL_GPIO_DeInit(GPIOA, GPIO_PIN_9|GPIO_PIN_10);
00276
       /* USER CODE BEGIN USART1_MspDeInit 1 */
00277
00278
00279
       /* USER CODE END USART1_MspDeInit 1 */
00280
00281
00282 }
```

# 4.6.2.8 HAL\_UART\_MspInit()

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

# **Parâmetros**

huart UART handle pointer

#### Valores Retornados

None

# USART1 GPIO Configuration PA9 ----> USART1\_TX PA10 ----> USART1\_RX

#### Definição na linha 222 do arquivo stm32f1xx\_hal\_msp.c.

```
00223 {
00224
         GPIO InitTypeDef GPIO InitStruct = {0}:
00225
         if (huart->Instance==USART1)
00226
00227
         /* USER CODE BEGIN USART1_MspInit 0 */
00228
00229
         /* USER CODE END USART1_MspInit 0 */
         /* Peripheral clock enable */
00230
           __HAL_RCC_USART1_CLK_ENABLE();
00231
00232
00233
             _HAL_RCC_GPIOA_CLK_ENABLE();
           GPIO_InitStruct.Pin = GPIO_PIN_9;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREO_HIGH;
00238
00239
00240
00241
           HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
00242
00243
           GPIO_InitStruct.Pin = GPIO_PIN_10;
          GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
00244
00245
00246
         HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
00247
00248
        /* USER CODE BEGIN USART1_MspInit 1 */
00249
00250
        /* USER CODE END USART1_MspInit 1 */
00251
00252
00253 }
```

# 4.7 stm32f1xx\_hal\_msp.c

#### Vá para a documentação desse arquivo.

```
00001 /* USER CODE BEGIN Header *
00020 /* USER CODE END Header */
00021
00022 /* Includes -
00023 #include "main.h"
00024 /* USER CODE BEGIN Includes */
00025
00026 /* USER CODE END Includes */
00027
00028 /* Private typedef -
00029 /* USER CODE BEGIN TD */
00030
00031 /* USER CODE END TD */
00032
00033 /* Private define -----*/
00034 /* USER CODE BEGIN Define */
00035
00036 /* USER CODE END Define */
00037
00038 /* Private macro -----
00039 /* USER CODE BEGIN Macro */
00040
00041 /* USER CODE END Macro */
00043 /* Private variables -----
00044 /* USER CODE BEGIN PV */
00045
00046 /* USER CODE END PV */
00047
00048 /* Private function prototypes -----*/
00049 /* USER CODE BEGIN PFP */
00050
00051 /* USER CODE END PFP */
00052
00053 /* External functions -----
00054 /* USER CODE BEGIN ExternalFunctions */
00056 /* USER CODE END ExternalFunctions */
```

```
00057
00058 /* USER CODE BEGIN 0 */
00059
00060 /* USER CODE END 0 */
00061
00062 void HAL_TIM_MspPostInit(TIM_HandleTypeDef *htim);
00066 void HAL_MspInit (void)
00067 {
00068
        /* USER CODE BEGIN MspInit 0 */
00069
00070
       /* USER CODE END MspInit 0 */
00071
00072
         _HAL_RCC_AFIO_CLK_ENABLE();
00073
        __HAL_RCC_PWR_CLK_ENABLE();
00074
00075
        /* System interrupt init*/
00076
        /* PendSV IROn interrupt configuration */
00077
        HAL_NVIC_SetPriority(PendSV_IRQn, 15, 0);
00078
00081
        __HAL_AFIO_REMAP_SWJ_DISABLE();
00082
00083
        /* USER CODE BEGIN MspInit 1 */
00084
00085
        /* USER CODE END MspInit 1 */
00086 }
00087
00094 void HAL_ADC_MspInit(ADC_HandleTypeDef* hadc)
00095 {
00096
        GPIO_InitTypeDef GPIO_InitStruct = {0};
00097
        if (hadc->Instance==ADC1)
00098
00099
        /* USER CODE BEGIN ADC1_MspInit 0 */
00100
00101
        /\star USER CODE END ADC1_MspInit 0 \star/
         /* Peripheral clock enable */
__HAL_RCC_ADC1_CLK_ENABLE();
00102
00103
00104
00105
            _HAL_RCC_GPIOA_CLK_ENABLE();
00109
          GPIO_InitStruct.Pin = GPIO_PIN_0;
00110
          GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
00111
          HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
00112
        /* USER CODE BEGIN ADC1_MspInit 1 */
00113
00114
00115
        /* USER CODE END ADC1_MspInit 1 */
00116
00117
00118 }
00119
00126 void HAL_ADC_MspDeInit(ADC_HandleTypeDef* hadc)
00127 {
00128
        if (hadc->Instance==ADC1)
00129
00130
        /* USER CODE BEGIN ADC1_MspDeInit 0 */
00131
00132
        /* USER CODE END ADC1 MspDeInit 0 */
00133
         /* Peripheral clock disable */
00134
          __HAL_RCC_ADC1_CLK_DISABLE();
00135
00139
          HAL_GPIO_DeInit (GPIOA, GPIO_PIN_0);
00140
00141
        /* USER CODE BEGIN ADC1_MspDeInit 1 */
00142
00143
        /* USER CODE END ADC1_MspDeInit 1 */
00144
00145
00146 }
00147
00154 void HAL_TIM_PWM_MspInit(TIM_HandleTypeDef* htim_oc)
00155 {
00156
        if (htim_oc->Instance==TIM1)
00157
00158
       /* USER CODE BEGIN TIM1_MspInit 0 */
00159
       /* USER CODE END TIM1_MspInit 0 */
00160
00161
        /* Peripheral clock enable */
00162
            _HAL_RCC_TIM1_CLK_ENABLE();
00163
        /* USER CODE BEGIN TIM1_MspInit 1 */
00164
00165
        /* USER CODE END TIM1 MspInit 1 */
00166
00167
00168 }
00169
00170 void HAL_TIM_MspPostInit(TIM_HandleTypeDef* htim)
00171 {
00172
        GPIO_InitTypeDef GPIO_InitStruct = {0};
```

```
if (htim->Instance==TIM1)
00174
00175
        /* USER CODE BEGIN TIM1_MspPostInit 0 */
00176
00177
        /* USER CODE END TIM1 MspPostInit 0 */
00178
00179
            _HAL_RCC_GPIOA_CLK_ENABLE();
00183
          GPIO_InitStruct.Pin = GPIO_PIN_8;
          GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
00184
00185
00186
00187
00188
        /* USER CODE BEGIN TIM1_MspPostInit 1 */
00189
00190
        /* USER CODE END TIM1_MspPostInit 1 */
00191
00192
00193 }
00200 void HAL_TIM_PWM_MspDeInit(TIM_HandleTypeDef* htim_oc)
00201 {
00202
        if (htim_oc->Instance==TIM1)
00203
00204
        /* USER CODE BEGIN TIM1 MspDeInit 0 */
00205
00206
       /* USER CODE END TIM1_MspDeInit 0 */
00207
        /* Peripheral clock disable */
00208
            _HAL_RCC_TIM1_CLK_DISABLE();
00209
        /* USER CODE BEGIN TIM1_MspDeInit 1 */
00210
        /* USER CODE END TIM1_MspDeInit 1 */
00211
00212
00213
00214 }
00215
00222 void HAL_UART_MspInit (UART_HandleTypeDef* huart)
00223 {
00224
        GPIO InitTypeDef GPIO InitStruct = {0};
        if (huart->Instance==USART1)
00226
00227
        /* USER CODE BEGIN USART1_MspInit 0 */
00228
        /* USER CODE END USART1_MspInit 0 */
00229
         /* Peripheral clock enable */
00230
00231
          __HAL_RCC_USART1_CLK_ENABLE();
00232
00233
            _HAL_RCC_GPIOA_CLK_ENABLE();
          GPIO_InitStruct.Pin = GPIO_PIN_9;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00238
00239
          GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
00240
00241
          HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
00242
00243
          GPIO_InitStruct.Pin = GPIO_PIN_10;
          GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
00244
00245
00246
          HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
00247
00248
        /* USER CODE BEGIN USART1_MspInit 1 */
00249
00250
        /* USER CODE END USART1_MspInit 1 */
00251
00252
00253 }
00254
00261 void HAL_UART_MspDeInit (UART_HandleTypeDef* huart)
00262 {
00263
        if (huart->Instance==USART1)
00264
00265
        /* USER CODE BEGIN USART1_MspDeInit 0 */
00266
00267
        /* USER CODE END USART1_MspDeInit 0 */
00268
         /* Peripheral clock disable */
00269
          __HAL_RCC_USART1_CLK_DISABLE();
00270
00275
          HAL_GPIO_DeInit(GPIOA, GPIO_PIN_9|GPIO_PIN_10);
00276
00277
        /* USER CODE BEGIN USART1_MspDeInit 1 */
00278
00279
        /* USER CODE END USART1_MspDeInit 1 */
00280
00281
00282 }
00283
00284 /* USER CODE BEGIN 1 */
00285
00286 /* USER CODE END 1 */
00287
```

# 4.8 Referência do Arquivo

D:/BACKUP/Faculdade/16\_Embarcados/Controlador\_Temperatura/
Controle-de-temperatura/Codigos/controle\_temperatura\_RTOS/
Core/Src/stm32f1xx hal timebase tim.c

HAL time base based on the hardware TIM.

```
#include "stm32f1xx_hal.h"
#include "stm32f1xx_hal_tim.h"
```

# **Funções**

HAL StatusTypeDef HAL InitTick (uint32 t TickPriority)

This function configures the TIM4 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.

#### **Variáveis**

• TIM\_HandleTypeDef htim4

# 4.8.1 Descrição detalhada

HAL time base based on the hardware TIM.

Atenção

© Copyright (c) 2021 STMicroelectronics. All rights reserved.

This software component is licensed by ST under Ultimate Liberty license SLA0044, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: www.st.com/SLA0044

Definição no arquivo stm32f1xx hal timebase tim.c.

# 4.8.2 Funções

# 4.8.2.1 HAL\_InitTick()

This function configures the TIM4 as a time base source. The time source is configured to have 1ms time base with a dedicated Tick interrupt priority.

#### Observação

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().

#### **Parâmetros**

TickPriority	Tick interrupt priority.

#### Valores Retornados

```
HAL status
```

Definição na linha 42 do arquivo stm32f1xx\_hal\_timebase\_tim.c.

```
00043
00044
         RCC_ClkInitTypeDef
                                  clkconfig;
00045
         uint32 t
                                   uwTimclock = 0;
00046
         uint32_t
                                   uwPrescalerValue = 0;
00047
                                  pFLatency;
00048
          /*Configure the TIM4 IRQ priority */
00049
         HAL_NVIC_SetPriority(TIM4_IRQn, TickPriority ,0);
00050
00051
         /* Enable the TIM4 global Interrupt */
00052
        HAL_NVIC_EnableIRQ(TIM4_IRQn);
00053
         /* Enable TIM4 clock *,
00054
         __HAL_RCC_TIM4_CLK_ENABLE();
00055
00056
         /* Get clock configuration */
         HAL_RCC_GetClockConfig(&clkconfig, &pFLatency);
00057
00058
00059
         /* Compute TIM4 clock */
00060
         uwTimclock = 2*HAL_RCC_GetPCLK1Freq();
        /* Compute the prescaler value to have TIM4 counter clock equal to 1MHz */ uwPrescalerValue = (uint32_t) ((uwTimclock / 1000000U) - 1U);
00061
00062
00063
00064
         /* Initialize TIM4 */
00065
         htim4.Instance = TIM4;
00066
00067
         /\star Initialize TIMx peripheral as follow:
         + Period = [(TIM4CLK/1000) - 1]. to have a (1/1000) s time base.
+ Prescaler = (uwTimclock/1000000 - 1) to have a 1MHz counter clock.
00068
00069
00070
         + ClockDivision = 0
00071
         + Counter direction = Up
00072
00073
         htim4.Init.Period = (1000000U / 1000U) - 1U;
00074
         htim4.Init.Prescaler = uwPrescalerValue;
00075
         htim4.Init.ClockDivision = 0;
htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
00076
00077
         if (HAL_TIM_Base_Init(&htim4) == HAL_OK)
00078
00079
            /\star Start the TIM time Base generation in interrupt mode \star/
08000
           return HAL_TIM_Base_Start_IT(&htim4);
00081
00082
00083
         /* Return function status */
00084
         return HAL_ERROR;
00085 }
```

4000		DagumaT	۱۰/۱۵:
4.8.2.2	ПAL	ResumeT	ICK

```
void HAL_ResumeTick (
     void )
```

Resume Tick increment.

Observação

Enable the tick increment by Enabling TIM4 update interrupt.

**Parâmetros** 

None

**Valores Retornados** 



Definição na linha 105 do arquivo stm32f1xx\_hal\_timebase\_tim.c.

# 4.8.2.3 HAL\_SuspendTick()

Suspend Tick increment.

Observação

Disable the tick increment by disabling TIM4 update interrupt.

**Parâmetros** 

None

**Valores Retornados** 

None

Definição na linha 93 do arquivo stm32f1xx\_hal\_timebase\_tim.c.

```
00094 {
00095    /* Disable TIM4 update Interrupt */
00096    __HAL_TIM_DISABLE_IT(&htim4, TIM_IT_UPDATE);
00097 }
```

#### 4.8.3 Variáveis

#### 4.8.3.1 htim4

TIM\_HandleTypeDef htim4

Definição na linha 29 do arquivo stm32f1xx\_hal\_timebase\_tim.c.

# 4.9 stm32f1xx hal timebase tim.c

# Vá para a documentação desse arquivo.

```
00001 /* USER CODE BEGIN Header */
00019 /* USER CODE END Header */
00020
00021 /* Includes -
00022 #include "stm32f1xx_hal.h"
00023 #include "stm32f1xx_hal_tim.h"
00024
00025 /* Private typedef -----
00026 /* Private define -----*/
00027 /* Private macro -----*/
00028 /* Private variables ------*/
00029 TIM_HandleTypeDef
                          htim4;
00030 /* Private function prototypes -----*/
00031 /* Private functions -----*/
00032
00042 HAL_StatusTypeDef HAL_InitTick(uint32_t TickPriority)
00043 {
      RCC_ClkInitTypeDef
00044
00045
       uint32_t
                          uwTimclock = 0;
00046
       uint32 t
                          uwPrescalerValue = 0;
00047
                           pFLatency;
       uint32 t
       /*Configure the TIM4 IRQ priority */
00048
       HAL_NVIC_SetPriority(TIM4_IRQn, TickPriority ,0);
00050
00051
       /\star Enable the TIM4 global Interrupt \star/
00052
      HAL_NVIC_EnableIRQ(TIM4_IRQn);
00053
      /* Enable TIM4 clock */
       __HAL_RCC_TIM4_CLK_ENABLE();
00054
00055
00056
       /* Get clock configuration */
00057
       HAL_RCC_GetClockConfig(&clkconfig, &pFLatency);
00058
00059
       /* Compute TIM4 clock */
      uwTimclock = 2*HAL_RCC_GetPCLK1Freq();
00060
00061
       /* Compute the prescaler value to have TIM4 counter clock equal to 1MHz */
00062
      uwPrescalerValue = (uint32_t) ((uwTimclock / 1000000U) - 1U);
00063
00064
       /* Initialize TIM4 */
00065
       htim4.Instance = TIM4;
00066
00067
       /* Initialize TIMx peripheral as follow:
       + Period = [(TIM4CLK/1000) - 1]. to have a (1/1000) s time base.
00068
00069
       + Prescaler = (uwTimclock/1000000 - 1) to have a 1MHz counter clock.
00070
       + ClockDivision = 0
00071
       + Counter direction = Up
00072
00073
       htim4.Init.Period = (1000000U / 1000U) - 1U;
       htim4.Init.Prescaler = uwPrescalerValue;
00075
       htim4.Init.ClockDivision = 0;
00076
       htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
00077
       if (HAL_TIM_Base_Init(&htim4) == HAL_OK)
00078
00079
        /\star Start the TIM time Base generation in interrupt mode \star/
        return HAL_TIM_Base_Start_IT(&htim4);
08000
00081
00082
00083
       /* Return function status */
00084
       return HAL_ERROR;
00085 }
00086
00093 void HAL_SuspendTick(void)
```

# 4.10 Referência do Arquivo D:/BACKUP/Faculdade/16\_Embarcados/← Controlador\_Temperatura/Controle-de-temperatura/← Codigos/controle temperatura RTOS/Core/Src/stm32f1xx it.c

Interrupt Service Routines.

```
#include "main.h"
#include "stm32f1xx_it.h"
```

# **Funções**

void NMI\_Handler (void)

This function handles Non maskable interrupt.

void HardFault\_Handler (void)

This function handles Hard fault interrupt.

• void MemManage\_Handler (void)

This function handles Memory management fault.

void BusFault Handler (void)

This function handles Prefetch fault, memory access fault.

void UsageFault\_Handler (void)

This function handles Undefined instruction or illegal state.

void DebugMon\_Handler (void)

This function handles Debug monitor.

• void TIM4\_IRQHandler (void)

This function handles TIM4 global interrupt.

# **Variáveis**

• TIM\_HandleTypeDef htim4

# 4.10.1 Descrição detalhada

Interrupt Service Routines.

Atenção

# © Copyright (c) 2021 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

Definição no arquivo stm32f1xx it.c.

# 4.10.2 Funções

# 4.10.2.1 BusFault Handler()

This function handles Prefetch fault, memory access fault.

```
Definição na linha 116 do arquivo stm32f1xx it.c.
```

# 4.10.2.2 DebugMon\_Handler()

This function handles Debug monitor.

#### Definição na linha 146 do arquivo stm32f1xx\_it.c.

#### 4.10.2.3 HardFault\_Handler()

This function handles Hard fault interrupt.

Definição na linha 86 do arquivo stm32f1xx it.c.

#### 4.10.2.4 MemManage Handler()

This function handles Memory management fault.

Definição na linha 101 do arquivo stm32f1xx\_it.c.

# 4.10.2.5 NMI\_Handler()

```
void NMI_Handler (
     void )
```

This function handles Non maskable interrupt.

Definição na linha 71 do arquivo stm32f1xx\_it.c.

#### 4.10.2.6 TIM4\_IRQHandler()

```
void TIM4_IRQHandler (
     void )
```

This function handles TIM4 global interrupt.

Definição na linha 166 do arquivo stm32f1xx it.c.

# 4.10.2.7 UsageFault\_Handler()

This function handles Undefined instruction or illegal state.

#### Definição na linha 131 do arquivo stm32f1xx it.c.

# 4.10.3 Variáveis

# 4.10.3.1 htim4

```
TIM_HandleTypeDef htim4 [extern]
```

Definição na linha 29 do arquivo stm32f1xx\_hal\_timebase\_tim.c.

4.11 stm32f1xx it.c 47

# 4.11 stm32f1xx\_it.c

```
Vá para a documentação desse arquivo.
00001 /* USER CODE BEGIN Header * 00019 /* USER CODE END Header */
00020
00021 /* Includes -
00022 #include "main.h"
00023 #include "stm32f1xx_it.h"
00024 /* Private includes -----
00025 /* USER CODE BEGIN Includes */
00026 /* USER CODE END Includes */
00028 /* Private typedef
00029 /* USER CODE BEGIN TD */
00030
00031 /* USER CODE END TD */
00032
00033 /* Private define -
00034 /* USER CODE BEGIN PD */
00035
00036 /* USER CODE END PD */
00037
00038 /* Private macro --
00039 /* USER CODE BEGIN PM */
00041 /* USER CODE END PM */
00042
00043 /* Private variables -----*/
00044 /* USER CODE BEGIN PV */
00045
00046 /* USER CODE END PV */
00048 /* Private function prototypes -----
00049 /* USER CODE BEGIN PFP */
00050
00051 /* USER CODE END PFP */
00052
00053 /* Private user code ---
00054 /* USER CODE BEGIN 0 */
00055
00056 /* USER CODE END 0 */
00057
00058 /* External variables --
00059 extern TIM_HandleTypeDef htim4;
00060
00061 /* USER CODE BEGIN EV */
00062
00063 /* USER CODE END EV */
00064
00066 /*
                 Cortex-M3 Processor Interruption and Exception Handlers
00071 void NMI_Handler(void)
00072 {
00073
      /* USER CODE BEGIN NonMaskableInt IROn 0 */
00075
       /* USER CODE END NonMaskableInt_IRQn 0 */
00076
       /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
00077
       while (1)
00078
00079
       /* USER CODE END NonMaskableInt_IRQn 1 */
00081 }
00082
00086 void HardFault_Handler(void)
00087 {
00088
       /* USER CODE BEGIN HardFault IROn 0 */
00089
00090
       /* USER CODE END HardFault_IRQn 0 */
       while (1)
00091
00092
       /* USER CODE BEGIN W1_HardFault_IRQn 0 */
/* USER CODE END W1_HardFault_IRQn 0 */
00093
00094
00095
00096 }
00097
00101 void MemManage_Handler(void)
00102 {
       /* USER CODE BEGIN MemoryManagement_IRQn 0 */
00103
00104
       /* USER CODE END MemoryManagement_IRQn 0 */
00106
       while (1)
00107
00108
         /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
```

```
/* USER CODE END W1_MemoryManagement_IRQn 0 */
00111 }
00112
00116 void BusFault Handler (void)
00117 {
00118
        /* USER CODE BEGIN BusFault_IRQn 0 */
00119
00120
       /* USER CODE END BusFault_IRQn 0 */
00121
        while (1)
00122
        /* USER CODE BEGIN W1_BusFault_IRQn 0 */
00123
00124
         /* USER CODE END W1_BusFault_IRQn 0 */
00125
00126 }
00127
00131 void UsageFault_Handler(void)
00132 {
00133
       /* USER CODE BEGIN UsageFault_IRQn 0 */
00135
        /* USER CODE END UsageFault_IRQn 0 */
00136
       while (1)
00137
         /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
00138
00139
         /* USER CODE END W1_UsageFault_IRQn 0 */
00141 }
00142
00146 void DebugMon_Handler(void)
00147 {
00148
       /* USER CODE BEGIN DebugMonitor IROn 0 */
00149
00150
       /* USER CODE END DebugMonitor_IRQn 0 */
00151
       /* USER CODE BEGIN DebugMonitor_IRQn 1 */
00152
00153
       /* USER CODE END DebugMonitor_IRQn 1 */
00154 }
00157 /* STM32F1xx Peripheral Interrupt Handlers
00158 /\star Add here the Interrupt Handlers for the used peripherals.
00159 /\star For the available peripheral interrupt handler names,
00160 /* please refer to the startup file (startup_stm32f1xx.s).
00161 /********
00166 void TIM4_IRQHandler(void)
00167 {
00168
       /* USER CODE BEGIN TIM4_IRQn 0 */
00169
00170
       /* USER CODE END TIM4_IRQn 0 */
      HAL_TIM_IRQHandler(&htim4);
00172
       /* USER CODE BEGIN TIM4_IRQn 1 */
00173
00174
       /* USER CODE END TIM4_IRQn 1 */
00175 }
00176
00177 /* USER CODE BEGIN 1 */
00178
00179 /* USER CODE END 1 */
00180 /********************************* (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

# 4.12 Referência do Arquivo D:/BACKUP/Faculdade/16\_Embarcados/ Controlador\_Temperatura/Controle-de-temperatura/ Codigos/controle temperatura RTOS/Core/Src/syscalls.c

STM32CubeIDE Minimal System calls file.

```
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
```

# **Funções**

- 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 <u>exit</u> (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 <u>\_times</u> (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)

# **Variáveis**

- · int errno
- char \*\* environ = \_\_env

# 4.12.1 Descrição detalhada

STM32CubeIDE Minimal System calls file.

Autor

Auto-generated by STM32CubeIDE

For more information about which c-functions need which of these lowlevel functions please consult the Newlib libc-manual

Atenção

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

Definição no arquivo syscalls.c.

# 4.12.2 Funções

# 4.12.2.1 \_\_attribute\_\_()

# Definição na linha 69 do arquivo syscalls.c.

```
00070 {
00071    int DataIdx;
00072
00073    for (DataIdx = 0; DataIdx < len; DataIdx++)
00074    {
00075         *ptr++ = __io_getchar();
00076    }
00077
00078    return len;
00079 }
```

# 4.12.2.2 \_\_io\_getchar()

# Definição na linha 39 do arquivo syscalls.c.

00043 { 0 };

# 4.12.2.3 \_close()

# Definição na linha 92 do arquivo syscalls.c.

```
00093 {
00094 return -1;
00095 }
```

# 4.12.2.4 \_execve()

# Definição na linha 155 do arquivo syscalls.c.

```
00156 {
00157 errno = ENOMEM;
00158 return -1;
00159 }
```

# 4.12.2.5 \_exit()

```
void _exit (
          int status )
```

# Definição na linha 63 do arquivo syscalls.c.

```
00064 {
00065    _kill(status, -1);
00066    while (1) {}    /* Make sure we hang here */
00067 }
```

# 4.12.2.6 \_fork()

```
int _fork (
     void )
```

# Definição na linha 149 do arquivo syscalls.c.

```
00150 {
00151 errno = EAGAIN;
00152 return -1;
00153 }
```

# 4.12.2.7 \_fstat()

# Definição na linha 98 do arquivo syscalls.c.

#### 4.12.2.8 getpid()

```
int _getpid (
          void )
```

# Definição na linha 52 do arquivo syscalls.c.

```
00053 {
00054 return 1;
00055 }
```

# 4.12.2.9 \_isatty()

```
int _isatty (
          int file )
```

# Definição na linha 104 do arquivo syscalls.c.

```
00105 {
00106 return 1;
00107 }
```

# 4.12.2.10 \_kill()

# Definição na linha 57 do arquivo syscalls.c.

```
00058 {
00059 errno = EINVAL;
00060 return -1;
00061 }
```

# 4.12.2.11 \_link()

```
int _link ( \label{char} \mbox{char} \ * \ old, \\ \mbox{char} \ * \ new \ )
```

# Definição na linha 143 do arquivo syscalls.c.

```
00144 {
00145 errno = EMLINK;
00146 return -1;
00147 }
```

# 4.12.2.12 \_lseek()

```
int _lseek (
                int file,
                int ptr,
                int dir )
```

# Definição na linha 109 do arquivo syscalls.c.

```
00110 {
00111 return 0;
00112 }
```

# 4.12.2.13 \_open()

#### Definição na linha 114 do arquivo syscalls.c.

# 4.12.2.14 \_stat()

#### Definição na linha 137 do arquivo syscalls.c.

# 4.12.2.15 \_times()

#### Definição na linha 132 do arquivo syscalls.c.

```
00133 {
00134 return -1;
00135 }
```

# 4.12.2.16 \_unlink()

#### Definição na linha 126 do arquivo syscalls.c.

```
00127 {
00128 errno = ENOENT;
00129 return -1;
00130 }
```

# 4.12.2.17 \_wait()

Definição na linha 120 do arquivo syscalls.c.

#### 4.12.2.18 initialise monitor handles()

```
void initialise_monitor_handles ( )
```

Definição na linha 48 do arquivo syscalls.c. 00049 { 00050 }

#### 4.12.3 Variáveis

#### 4.12.3.1 environ

```
char** environ = __env
```

Definição na linha 44 do arquivo syscalls.c.

# 4.13 syscalls.c

#### Vá para a documentação desse arquivo.

```
00024 /* Includes */
00025 #include <sys/stat.h>
00026 #include <stdlib.h>
00027 #include <errno.h>
00028 #include <stdio.h>
00029 #include <signal.h>
00030 #include <time.h>
00031 #include <sys/time.h>
00032 #include <sys/times.h>
00033
00034
00035 /* Variables */
00036 //#undef errno
00037 extern int errno;
00038 extern int __io_putchar(int ch) __attribute__((weak));
00039 extern int __io_getchar(void) __attribute__((weak));
00040
00041 register char * stack_ptr asm("sp");
00042
00043 char *__env[1] = { 0 };
00044 char **environ = __env;
00045
00046
00047 /* Functions */
00048 void initialise_monitor_handles()
00049 {
```

4.13 syscalls.c 55

```
00050 }
00051
00052 int _getpid(void)
00053 {
00054
          return 1;
00055 }
00057 int _kill(int pid, int sig)
00058 {
         errno = EINVAL;
return -1;
00059
00060
00061 }
00062
00063 void _exit (int status)
00064 {
00065
          _kill(status, -1);
                              /* Make sure we hang here */
00066
          while (1) {}
00067 }
00068
00069 __attribute__((weak)) int _read(int file, char *ptr, int len)
00070 {
00071
          int DataIdx;
00072
00073
          for (DataIdx = 0; DataIdx < len; DataIdx++)</pre>
00074
        {
00075
              *ptr++ = ___io_getchar();
00076
00077
00078 return len;
00079 }
08000
00081 <u></u>
       _attribute__((weak)) int _write(int file, char *ptr, int len)
00083
          int DataIdx;
00084
          for (DataIdx = 0; DataIdx < len; DataIdx++)</pre>
00085
00086
         {
00087
              __io_putchar(*ptr++);
00088
00089
          return len;
00090 }
00091
00092 int _close(int file)
00093 {
00094
          return -1;
00095 }
00096
00097
00098 int _fstat(int file, struct stat *st)
00099 {
00100
         st->st_mode = S_IFCHR;
00101
         return 0;
00102 }
00103
00104 int _isatty(int file)
00105 {
          return 1;
00107 }
00108
00109 int _lseek(int file, int ptr, int dir)
00110 {
00111
          return 0;
00112 }
00113
00114 int _open(char *path, int flags, ...)
00115 {
00116
         /* Pretend like we always fail */
         return -1;
00117
00118 }
00119
00120 int _wait(int *status)
00121 {
         errno = ECHILD;
return -1;
00122
00123
00124 }
00125
00126 int _unlink(char *name)
00127 {
         errno = ENOENT;
00128
00129
          return -1:
00130 }
00131
00132 int _times(struct tms *buf)
00133 {
00134
          return -1;
00135 }
00136
```

```
00137 int _stat(char *file, struct stat *st)
00139
          st->st_mode = S_IFCHR;
00140
          return 0;
00141 }
00142
00143 int _link(char *old, char *new)
00144 {
00145
          errno = EMLINK;
00146
          return -1;
00147 }
00148
00149 int _fork(void)
00150 {
00151
          errno = EAGAIN;
00152
          return -1;
00153 }
00154
00155 int _execve(char *name, char **argv, char **env)
00156 {
          errno = ENOMEM;
return -1;
00157
00158
00159 }
```

# 4.14 Referência do Arquivo D:/BACKUP/Faculdade/16\_Embarcados/ Controlador\_Temperatura/Controle-de-temperatura/ Codigos/controle temperatura RTOS/Core/Src/sysmem.c

STM32CubeIDE System Memory calls file.

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

# **Funções**

void \* \_sbrk (ptrdiff\_t incr)
 \_sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

# 4.14.1 Descrição detalhada

STM32CubeIDE System Memory calls file.

**Autor** 

# Generated by STM32CubeIDE

For more information about which C functions need which of these lowlevel functions please consult the newlib libc manual

Atenção

#### © Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

Definição no arquivo sysmem.c.

# 4.14.2 Funções

#### 4.14.2.1 sbrk()

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

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'.

#### **Parâmetros**

```
incr | Memory size
```

#### Retorna

Pointer to allocated memory

#### Definição na linha 54 do arquivo sysmem.c.

```
00055 {
00056
          extern uint8_t _end; /* Symbol defined in the linker script */
00057
          extern uint8_t _estack; /* Symbol defined in the linker script */
         extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker script */
const uint32_t stack_limit = (uint32_t)&_estack - (uint32_t)&_Min_Stack_Size;
const uint8_t *max_heap = (uint8_t *) stack_limit;
00058
00059
00060
00061
         uint8_t *prev_heap_end;
00062
00063
          /\star Initialize heap end at first call \star/
00064
         if (NULL == __sbrk_heap_end)
00065 {
00066
               _sbrk_heap_end = &_end;
00067
```

```
00069
        /\star Protect heap from growing into the reserved MSP stack \star/
00070
        if (__sbrk_heap_end + incr > max_heap)
00071
       errno = ENOMEM;
00072
00073
         return (void *)-1;
00075
00076
       prev_heap_end = __sbrk_heap_end;
00077
       __sbrk_heap_end += incr;
00078
00079
       return (void *)prev_heap_end;
00080 }
```

# 4.15 sysmem.c

Vá para a documentação desse arquivo.

```
00024 /* Includes */
00025 #include <errno.h>
00026 #include <stdint.h>
00027
00031 static uint8_t *__sbrk_heap_end = NULL;
00032
00054 void *_sbrk(ptrdiff_t incr)
00055 {
00056 extern uint8_t _end; /* Symbol defined in the linker script */
        extern uint8_t _estack; /* Symbol defined in the linker script */
extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker script */
const uint32_t stack_limit = (uint32_t)&_estack - (uint32_t)&_Min_Stack_Size;
00057
00058
00059
00060
         const uint8_t *max_heap = (uint8_t *)stack_limit;
00061
         uint8_t *prev_heap_end;
00062
00063
         /\star Initialize heap end at first call \star/
00064
         if (NULL == __sbrk_heap_end)
00065
00066
              sbrk heap end = & end;
00067
00068
00069
         /\star Protect heap from growing into the reserved MSP stack \star/
00070
         if (__sbrk_heap_end + incr > max_heap)
00071
         errno = ENOMEM;
00072
           return (void *)-1;
00074
00075
        prev_heap_end = __sbrk_heap_end;
__sbrk_heap_end += incr;
00076
00077
00078
         return (void *)prev_heap_end;
```

# 4.16 Referência do Arquivo

D:/BACKUP/Faculdade/16\_Embarcados/Controlador\_Temperatura/← Controle-de-temperatura/Codigos/controle\_temperatura\_RTOS/← Core/Src/system\_stm32f1xx.c

CMSIS Cortex-M3 Device Peripheral Access Layer System Source File.

```
#include "stm32f1xx.h"
```

# Definições e Macros

- #define HSE VALUE 8000000U
- #define HSI\_VALUE 8000000U

#### **Funções**

void SystemInit (void)

Setup the microcontroller system Initialize the Embedded Flash Interface, the PLL and update the SystemCoreClock variable.

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.

#### **Variáveis**

- uint32\_t SystemCoreClock = 16000000
- const uint8\_t AHBPrescTable [16U] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8 t APBPrescTable [8U] = {0, 0, 0, 0, 1, 2, 3, 4}

# 4.16.1 Descrição detalhada

CMSIS Cortex-M3 Device Peripheral Access Layer System Source File.

**Autor** 

MCD Application Team

- 1. This file provides two functions and one global variable to be called from user application:
  - SystemInit(): Setups the system clock (System clock source, PLL Multiplier factors, AHB/APBx prescalers and Flash settings). This function is called at startup just after reset and before branch to main program. This call is made inside the "startup\_stm32f1xx\_xx.s" file.
  - SystemCoreClock variable: Contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.
  - SystemCoreClockUpdate(): Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.
- 2. After each device reset the HSI (8 MHz) is used as system clock source. Then SystemInit() function is called, in "startup\_stm32f1xx\_xx.s" file, to configure the system clock before to branch to main program.
- 3. The default value of HSE crystal is set to 8 MHz (or 25 MHz, depending on the product used), refer to "HSE ← \_VALUE". When HSE is used as system clock source, directly or through PLL, and you are using different crystal you have to adapt the HSE value to your own configuration.

Atenção

#### © Copyright (c) 2017 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

Definição no arquivo system\_stm32f1xx.c.

# 4.17 system stm32f1xx.c

#### Vá para a documentação desse arquivo.

```
00001
00059 #include "stm32f1xx.h"
00060
00077 #if !defined (HSE_VALUE)
        #define HSE_VALUE
00078
                                           8000000U
00080 #endif /* HSE_VALUE */
00081
00082 #if !defined (HSI VALUE)
        #define HSI_VALUE
                                           800000011
00083
00085 #endif /* HSI_VALUE */
00088 #if defined(STM32F100xE) || defined(STM32F101xE) || defined(STM32F101xG) || defined(STM32F103xE) ||
       defined(STM32F103xG)
00089 /* #define DATA_IN_ExtSRAM */
00090 #endif /* STM32F100xE || STM32F101xE || STM32F101xG || STM32F103xE || STM32F103xG */
00091
00092 /\star Note: Following vector table addresses must be defined in line with linker
00093
               configuration. */
00097 /* #define USER_VECT_TAB_ADDRESS */
00098
00099 #if defined(USER VECT TAB ADDRESS)
00102 /* #define VECT_TAB_SRAM */
00103 #if defined(VECT_TAB_SRAM)
00104 #define VECT_TAB_BASE_ADDRESS
                                        SRAM BASE
00106 #define VECT_TAB_OFFSET
                                        0x00000000U
00108 #else
00109 #define VECT_TAB_BASE_ADDRESS FLASH_BASE
00111 #define VECT_TAB_OFFSET 0x000000000
00113 #endif /* VECT_TAB_SRAM */
                                        0x00000000U
00114 #endif /* USER_VECT_TAB_ADDRESS */
00115
00116 /*****************************
00117
        /* This variable is updated in three ways:
1) by calling CMSIS function SystemCoreClockUpdate()
00134
00135
             2) by calling HAL API function HAL_RCC_GetHCLKFreq()
00136
00137
             3) each time HAL_RCC_ClockConfig() is called to configure the system clock frequency
               Note: If you use this function to configure the system clock; then there is no need to call the 2 first functions listed above, since SystemCoreClock
00138
00139
00140
                      variable is updated automatically.
00141
00142 uint32_t SystemCoreClock = 16000000;
00143 const uint8_t AHBPrescTable[16U] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9};
00144 const uint8_t APBPrescTable[8U] = {0, 0, 0, 0, 1, 2, 3, 4};
00145
00154 #if defined(STM32F100xE) || defined(STM32F101xE) || defined(STM32F101xE) ||
       defined(STM32F103xG)
00155 #ifdef DATA_IN_ExtSRAM
00156
        static void SystemInit_ExtMemCtl(void);
00157 #endif /* DATA_IN_ExtSRAM */
00158 #endif /* STM32F100xE || STM32F101xE || STM32F101xG || STM32F103xE || STM32F103xG */
00159
00176 void SystemInit (void)
00177
00178 #if defined(STM32F100xE) || defined(STM32F101xE) || defined(STM32F101xG) || defined(STM32F103xE) ||
      defined(STM32F103xG)
00179
       #ifdef DATA_IN_ExtSRAM
00180
          SystemInit_ExtMemCtl();
00181
        #endif /* DATA_IN_ExtSRAM */
00182 #endif
00183
        /* Configure the Vector Table location -----*/
00184
00185 #if defined(USER_VECT_TAB_ADDRESS)
00186
       SCB->VTOR = VECT_TAB_BASE_ADDRESS | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM. */
00187 #endif /* USER_VECT_TAB_ADDRESS */
00188 }
00225 void SystemCoreClockUpdate (void)
00226 {
00227
        uint32_t tmp = 0U, pllmull = 0U, pllsource = 0U;
00228
00229 #if defined(STM32F105xC) || defined(STM32F107xC)
        uint32_t prediv1source = 0U, prediv1factor = 0U, prediv2factor = 0U, pl12mul1 = 0U;
00231 #endif /* STM32F105xC */
00232
00233 #if defined(STM32F100xB) || defined(STM32F100xE) 00234 uint32_t prediv1factor = 0U;
00235 #endif /* STM32F100xB or STM32F100xE */
00236
00237
         /* Get SYSCLK source --
00238
        tmp = RCC->CFGR & RCC_CFGR_SWS;
00239
```

```
00240
        switch (tmp)
00241
00242
          case 0x00U: /* HSI used as system clock */
            SystemCoreClock = HSI_VALUE;
00243
00244
            break;
00245
          case 0x04U: /* HSE used as system clock */
           SystemCoreClock = HSE_VALUE;
00247
00248
          case 0x08U: /* PLL used as system clock */
00249
00250
            /* Get PLL clock source and multiplication factor -----*/
            pllmull = RCC->CFGR & RCC_CFGR_PLLMULL;
00251
00252
            pllsource = RCC->CFGR & RCC_CFGR_PLLSRC;
00253
00254 #if !defined(STM32F105xC) && !defined(STM32F107xC)
00255
            pllmull = ( pllmull » 18U) + 2U;
00256
00257
             if (pllsource == 0x00U)
00258
00259
               /* HSI oscillator clock divided by 2 selected as PLL clock entry */
00260
               SystemCoreClock = (HSI_VALUE » 1U) * pllmull;
00261
00262
            else
00263
00264 #if defined(STM32F100xB) || defined(STM32F100xE)
             prediv1factor = (RCC->CFGR2 & RCC_CFGR2_PREDIV1) + 1U;
00266
              /\star HSE oscillator clock selected as PREDIV1 clock entry \star/
00267
              SystemCoreClock = (HSE_VALUE / prediv1factor) * pllmull;
00268 #else
00269
               /* HSE selected as PLL clock entry */
00270
               if ((RCC->CFGR & RCC_CFGR_PLLXTPRE) != (uint32_t)RESET)
00271
              {/* HSE oscillator clock divided by 2 */
00272
                SystemCoreClock = (HSE_VALUE » 1U) * pllmull;
00273
00274
               else
00275
00276
                SystemCoreClock = HSE_VALUE * pllmull;
00277
00278 #endif
00279
00280 #else
            pllmull = pllmull » 18U;
00281
00282
00283
             if (pllmull != 0x0DU)
00284
            {
00285
               pllmull += 2U;
00286
00287
             else
            { /* PLL multiplication factor = PLL input clock * 6.5 */
00288
              pllmull = 13U / 2U;
00289
00290
00291
00292
             if (pllsource == 0x00U)
00293
00294
               /* HSI oscillator clock divided by 2 selected as PLL clock entry */
00295
               SystemCoreClock = (HSI_VALUE » 1U) * pllmull;
00296
00297
00298
             {/* PREDIV1 selected as PLL clock entry */}
00299
00300
               /\star Get PREDIV1 clock source and division factor \star/
               prediv1source = RCC->CFGR2 & RCC_CFGR2_PREDIV1SRC;
00301
00302
               prediv1factor = (RCC->CFGR2 & RCC_CFGR2_PREDIV1) + 1U;
00303
00304
               if (prediv1source == 0U)
00305
               {
                /* HSE oscillator clock selected as PREDIV1 clock entry */
SystemCoreClock = (HSE_VALUE / prediv1factor) * pllmull;
00306
00307
00308
00309
00310
               {/* PLL2 clock selected as PREDIV1 clock entry */
00311
                 /* Get PREDIV2 division factor and PLL2 multiplication factor */ prediv2factor = ((RCC->CFGR2 & RCC_CFGR2_PREDIV2) » 4U) + 1U; pll2mull = ((RCC->CFGR2 & RCC_CFGR2_PLL2MUL) » 8U) + 2U;
00312
00313
00314
                 SystemCoreClock = (((HSE_VALUE / prediv2factor) * pl12mull) / prediv1factor) * pl1mull;
00315
00316
00317
            }
00318 #endif /* STM32F105xC */
00319
            break;
00320
00321
          default:
00322
            SystemCoreClock = HSI_VALUE;
00323
            break;
00324
        }
00325
```

```
/* Compute HCLK clock frequency -----*/
00327
       /* Get HCLK prescaler */
       tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) » 4U)];
00328
       /* HCLK clock frequency */
00329
00330
       SystemCoreClock >= tmp;
00331 }
00332
00333 #if defined(STM32F100xE) || defined(STM32F101xE) || defined(STM32F101xG) || defined(STM32F103xE) ||
      defined(STM32F103xG)
00340 #ifdef DATA_IN_ExtSRAM
00350 void SystemInit_ExtMemCtl(void)
00351 {
00352
        IO uint32 t tmpreq;
       /* Enable FSMC clock
00356
00357
       RCC->AHBENR = 0x00000114U;
00358
       /* Delay after an RCC peripheral clock enabling */
00359
      tmpreg = READ_BIT(RCC->AHBENR, RCC_AHBENR_FSMCEN);
00360
00361
00362
       /* Enable GPIOD, GPIOE, GPIOF and GPIOG clocks */
00363
       RCC->APB2ENR = 0x000001E0U;
00364
      /* Delay after an RCC peripheral clock enabling */
tmpreg = READ_BIT(RCC->APB2ENR, RCC_APB2ENR_IOPDEN);
00365
00366
00367
00368
       (void) (tmpreg);
00369
00370 /\star -----\star/ SRAM Data lines, NOE and NWE configuration -----\star/
00374 /*----- NBL0, NBL1 configuration -----*/
00375
00376
       GPIOD -> CRL = 0x44BB44BBU;
      GPIOD->CRH = 0xBBBBBBBBBU;
00377
00378
00379
       GPIOE \rightarrow CRL = 0xB44444BBU;
      GPIOE->CRH = 0xBBBBBBBBBU;
00380
00381
00382
       GPIOF->CRL = 0x44BBBBBBU;
      GPIOF->CRH = 0xBBBB4444U;
00383
00384
00385
       GPTOG \rightarrow CRI_{L} = 0 \times 44BBBBBBBIJ:
00386
      GPIOG->CRH = 0x444B4B44U;
00387
00388 /*--
           ----- FSMC Configuration -----
00389 /*----* Enable FSMC Bank1_SRAM Bank -----*/
00390
      FSMC Bank1->BTCR[4U] = 0 \times 000001091U;
00391
      FSMC_Bank1->BTCR[5U] = 0x00110212U;
00392
00393 }
00394 #endif /* DATA_IN_ExtSRAM */
00395 #endif /* STM32F100xE || STM32F101xE || STM32F101xG || STM32F103xE || STM32F103xG */ ^{\star}
00396
```

# **Índice Remissivo**

```
attribute
                                                                                                                                                                                                                                                                                  CSdis
                        syscalls.c, 50
                                                                                                                                                                                                                                                                                                           main.c, 16
          _io_getchar
                                                                                                                                                                                                                                                                                   CSen
                        syscalls.c, 50
                                                                                                                                                                                                                                                                                                          main.c, 16
  _close
                                                                                                                                                                                                                                                                                  D:/BACKUP/Faculdade/16_Embarcados/Controlador_Temperatura/Control
                        syscalls.c, 50
                                                                                                                                                                                                                                                                                                                                  de-temperatura/Codigos/controle temperatura RTOS/Core/Src/
  execve
                                                                                                                                                                                                                                                                                                                                  11.13
                        syscalls.c, 50
                                                                                                                                                                                                                                                                                  D:/BACKUP/Faculdade/16_Embarcados/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperat
 _exit
                                                                                                                                                                                                                                                                                                                                  de-temperatura/Codigos/controle_temperatura_RTOS/Core/Src/
                        syscalls.c, 50
 _fork
                                                                                                                                                                                                                                                                                  D:/BACKUP/Faculdade/16_Embarcados/Controlador_Temperatura/Control
                        syscalls.c, 51
                                                                                                                                                                                                                                                                                                                                  de-temperatura/Codigos/controle_temperatura_RTOS/Core/Src/
_fstat
                                                                                                                                                                                                                                                                                                                                   14, 26
                        syscalls.c, 51
                                                                                                                                                                                                                                                                                  D:/BACKUP/Faculdade/16_Embarcados/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperat
  _getpid
                                                                                                                                                                                                                                                                                                                                  de-temperatura/Codigos/controle_temperatura_RTOS/Core/Src/
                        syscalls.c, 51
 _isatty
                                                                                                                                                                                                                                                                                   D:/BACKUP/Faculdade/16 Embarcados/Controlador Temperatura/Control
                        syscalls.c, 51
                                                                                                                                                                                                                                                                                                                                  de-temperatura/Codigos/controle temperatura RTOS/Core/Src/
 _kill
                                                                                                                                                                                                                                                                                                                                  39, 42
                        syscalls.c, 52
                                                                                                                                                                                                                                                                                  D:/BACKUP/Faculdade/16 Embarcados/Controlador Temperatura/Control
 _link
                                                                                                                                                                                                                                                                                                                                  de-temperatura/Codigos/controle_temperatura_RTOS/Core/Src/
                        syscalls.c, 52
                                                                                                                                                                                                                                                                                                                                  43, 47
 _lseek
                                                                                                                                                                                                                                                                                  D:/BACKUP/Faculdade/16_Embarcados/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperat
                        syscalls.c, 52
                                                                                                                                                                                                                                                                                                                                  de-temperatura/Codigos/controle_temperatura_RTOS/Core/Src/
_open
                        syscalls.c, 52
                                                                                                                                                                                                                                                                                  D:/BACKUP/Faculdade/16_Embarcados/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperatura/Controlador_Temperat
  sbrk
                                                                                                                                                                                                                                                                                                                                  de-temperatura/Codigos/controle_temperatura_RTOS/Core/Src/
                        sysmem.c, 57
                                                                                                                                                                                                                                                                                                                                  56, 58
_stat
                                                                                                                                                                                                                                                                                   D:/BACKUP/Faculdade/16 Embarcados/Controlador Temperatura/Control
                        syscalls.c, 53
                                                                                                                                                                                                                                                                                                                                  de-temperatura/Codigos/controle_temperatura_RTOS/Core/Src/
  _times
                                                                                                                                                                                                                                                                                                                                  58,60
                        syscalls.c, 53
                                                                                                                                                                                                                                                                                   DebugMon_Handler
 _unlink
                                                                                                                                                                                                                                                                                                          stm32f1xx_it.c, 44
                        syscalls.c, 53
                                                                                                                                                                                                                                                                                   Display_Task
 _wait
                                                                                                                                                                                                                                                                                                          main.c, 23
                       syscalls.c, 53
                                                                                                                                                                                                                                                                                   Display_taskF
AHBPrescTable
                                                                                                                                                                                                                                                                                                          main.c, 18
                        STM32F1xx_System_Private_Variables, 6
                                                                                                                                                                                                                                                                                   dutyCycle
APBPrescTable
                                                                                                                                                                                                                                                                                                          main.c, 24
                        STM32F1xx System Private Variables, 6
                                                                                                                                                                                                                                                                                  environ
BusFault Handler
                                                                                                                                                                                                                                                                                                          syscalls.c, 54
                        stm32f1xx it.c, 44
                                                                                                                                                                                                                                                                                   Error Handler
                                                                                                                                                                                                                                                                                                          main.c, 19
CMSIS, 5
Control Task
                                                                                                                                                                                                                                                                                   Filter Task
                        main.c, 23
                                                                                                                                                                                                                                                                                                          main.c, 24
Control_taskF
                                                                                                                                                                                                                                                                                   Filter_taskF
                        main.c, 17
                                                                                                                                                                                                                                                                                                          main.c, 20
```

64 ÍNDICE REMISSIVO

filteredTemp	main
main.c, 24	main.c, 21
filteredTempQueue	main.c
main.c, 24	Control_Task, 23
FIRFilter.c	Control_taskF, 17
FIRFilter_Init, 11	CSdis, 16
FIRFilter_Update, 12	CSen, 16
FIRFilter_Init	Display_Task, 23
FIRFilter.c, 11	Display_taskF, 18
FIRFilter_Update	dutyCycle, 24
FIRFilter.c, 12	Error_Handler, 19
	Filter_Task, 24
hadc1	Filter_taskF, 20
main.c, 24	filteredTemp, 24
HAL_ADC_MspDeInit	filteredTempQueue, 24
stm32f1xx_hal_msp.c, 31	hadc1, 24
HAL_ADC_MspInit	HAL_TIM_PeriodElapsedCallback, 20
stm32f1xx_hal_msp.c, 32	htim1, 24
HAL_InitTick	huart1, 25
stm32f1xx_hal_timebase_tim.c, 39	k, 16
HAL_MspInit	ksat, 16
stm32f1xx_hal_msp.c, 33	main, 21
HAL_ResumeTick	p, 16
stm32f1xx_hal_timebase_tim.c, 40	r, 17
HAL SuspendTick	ref, 25
stm32f1xx_hal_timebase_tim.c, 41	SCK H, 17
HAL TIM MspPostInit	SCK L, 17
stm32f1xx_hal_msp.c, 33	SystemClock_Config, 22
HAL_TIM_PeriodElapsedCallback	Temp Task, 25
main.c, 20	Temp_taskF, 22
HAL_TIM_PWM_MspDeInit	tempFilter, 25
stm32f1xx_hal_msp.c, 33	•
HAL TIM PWM Msplnit	tempQueue, 25
stm32f1xx_hal_msp.c, 34	MemManage_Handler
HAL_UART_MspDeInit	stm32f1xx_it.c, 45
stm32f1xx_hal_msp.c, 35	NMI Handler
HAL_UART_MspInit	stm32f1xx_it.c, 45
stm32f1xx_hal_msp.c, 35	Stillo211xx_it.c, 45
HardFault Handler	p
stm32f1xx it.c, 44	main.c, 16
HSE_VALUE	mamo, ro
STM32F1xx_System_Private_Defines, 6	r
_ ·	main.c, 17
HSI_VALUE	ref
STM32F1xx_System_Private_Defines, 6	main.c, 25
htim1	
main.c, 24	SCK_H
htim4	main.c, 17
stm32f1xx_hal_timebase_tim.c, 42	SCK L
stm32f1xx_it.c, 46	main.c, 17
huart1	stm32f1xx_hal_msp.c
main.c, 25	HAL_ADC_MspDeInit, 31
	HAL_ADC_MspInit, 32
initialise_monitor_handles	HAL_MspInit, 33
syscalls.c, 54	HAL_TIM_MspPostInit, 33
L.	HAL_TIM_PWM_MspDeInit, 33
k main a 16	HAL_TIM_PWM_MspInit, 34
main.c, 16	HAL_UART_MspDeInit, 35
ksat	HAL_UART_MspInit, 35
main.c, 16	HAL_OAHH_IVIƏPIHIL, 33

ÍNDICE REMISSIVO 65

```
stm32f1xx_hal_timebase_tim.c
                                                           STM32F1xx_System_Private_Functions, 9
    HAL_InitTick, 39
                                                       Temp_Task
    HAL_ResumeTick, 40
                                                           main.c, 25
    HAL_SuspendTick, 41
                                                       Temp taskF
    htim4, 42
                                                           main.c, 22
stm32f1xx it.c
                                                      tempFilter
    BusFault Handler, 44
                                                           main.c, 25
     DebugMon Handler, 44
                                                      tempQueue
    HardFault Handler, 44
                                                           main.c, 25
    htim4, 46
                                                      TIM4 IRQHandler
    MemManage_Handler, 45
                                                           stm32f1xx it.c, 45
    NMI_Handler, 45
    TIM4_IRQHandler, 45
                                                      UsageFault_Handler
     UsageFault_Handler, 46
                                                           stm32f1xx_it.c, 46
Stm32f1xx_system, 5
STM32F1xx_System_Private_Defines, 5
    HSE VALUE, 6
    HSI VALUE, 6
STM32F1xx_System_Private_FunctionPrototypes, 7
STM32F1xx_System_Private_Functions, 7
     SystemCoreClockUpdate, 7
     SystemInit, 9
STM32F1xx_System_Private_Includes, 5
STM32F1xx_System_Private_Macros, 6
STM32F1xx_System_Private_TypesDefinitions, 5
STM32F1xx_System_Private_Variables, 6
    AHBPrescTable, 6
    APBPrescTable, 6
    SystemCoreClock, 7
syscalls.c
      _attribute__, 50
     __io_getchar, 50
    _close, 50
    _execve, 50
    _exit, 50
    _fork, 51
    _fstat, 51
    _getpid, 51
     _isatty, 51
     kill, 52
    _link, 52
    _lseek, 52
    _open, 52
    stat, 53
    _times, 53
    _unlink, 53
     _wait, <mark>53</mark>
    environ, 54
    initialise monitor handles, 54
sysmem.c
     _sbrk, <u>57</u>
SystemClock Config
    main.c, 22
SystemCoreClock
     STM32F1xx_System_Private_Variables, 7
SystemCoreClockUpdate
     STM32F1xx_System_Private_Functions, 7
SystemInit
```