Sistema de Monitoramento de Consumo de Energia

v1.0

Generated by Doxygen 1.8.13

Contents

1 Module Index

1.1 Modules

Here is a list of all modules:

2 Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

4 Module Documentation

4.1 CMSIS

Modules

- Stm32f4xx_system

4.1.1 Detailed Description

4.2 Stm32f4xx_system

Modules

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

4.2.1 Detailed Description

4.3 STM32F4xx_System_Private_Includes

Macros

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

4.3.1 Detailed Description

4.3.2 Macro Definition Documentation

4.3.2.1 HSE_VALUE

```
#define HSE_VALUE ((uint32_t)25000000)
```

Default value of the External oscillator in Hz

4.3.2.2 HSI_VALUE

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

Value of the Internal oscillator in Hz

4.5 STM32F4xx_System_Private_Defines

Macros

- #define VECT_TAB_OFFSET 0x00

4.5.1 Detailed Description

4.5.2 Macro Definition Documentation

4.5.2.1 VECT_TAB_OFFSET

```
#define VECT_TAB_OFFSET 0x00
```

< Uncomment the following line if you need to use external SRAM or SDRAM as data memory

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

4.7 STM32F4xx_System_Private_Variables

Variables

- uint32_t SystemCoreClock = 16000000
- const uint8_t AHBPrescTable [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t APBPrescTable [8] = {0, 0, 0, 0, 1, 2, 3, 4}

4.7.1 Detailed Description

4.7.2 Variable Documentation

4.7.2.1 AHBPrescTable

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

4.7.2.2 APBPrescTable

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

4.7.2.3 SystemCoreClock

```
uint32_t SystemCoreClock = 16000000
```

4.9 STM32F4xx_System_Private_Functions

Functions

- void SystemInit (void)

  Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.
- void SystemCoreClockUpdate (void)

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

4.9.1 Detailed Description

4.9.2 Function Documentation

4.9.2.1 SystemCoreClockUpdate()

```
void SystemCoreClockUpdate (
            void  )
```

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

Note

Each time the core clock (HCLK) changes, this function must be called to update SystemCoreClock variable value. Otherwise, any configuration based on this variable will be incorrect.
- The system frequency computed by this function is not the real frequency in the chip. It is calculated based on the predefined constant and the selected clock source:

- If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(∗)
- If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(∗∗)
- If SYSCLK source is PLL, SystemCoreClock will contain the HSE_VALUE(∗∗) or HSI_VALUE(∗) multiplied/divided by the PLL factors.

(∗) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h file (default value 16 MHz) but the real value may vary depending on the variations in voltage and temperature.

(∗∗) HSE_VALUE is a constant defined in stm32f4xx_hal_conf.h file (its value depends on the application requirements), user has to ensure that HSE_VALUE is same as the real frequency of the crystal used. Otherwise, this function may have wrong result.

- The result of this function could be not correct when using fractional value for HSE crystal.

Parameters

None	

Return values

None	

4.9.2.2 SystemInit()

```
void SystemInit (
             void  )
```

Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.

Parameters

None	

Return values

None	

5 Data Structure Documentation

5.1 Equipamento Struct Reference

Estrutura da base de dados de equipamentos.

`#include <equipamentos.h>`

Data Fields

- uint16_t ID
- char name [32]
- Parametros med

5.1.1 Detailed Description

Estrutura da base de dados de equipamentos.

Essa estrutura é uma entrada da base de dados de equipamentos. Armazena informações referentes a cada equipamento que pode ser identificado.

5.1.2 Field Documentation

5.1.2.1 ID

`uint16_t ID`

ID para o equipamento

5.1.2.2 med

`Parametros med`

Entrada para estrutura Parametros que armazena parâmetros elétricos de interesse.

5.1.2.3 name

`char name[32]`

String com o nome do equipamento.

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

- Inc/equipamentos.h

5.2 Medicao Struct Reference

Estrutura armazenada na memória com histórico de medições e equipamentos.

`#include <equipamentos.h>`

Data Fields

- uint32_t timestamp
- uint16_t equipamentos [EQUIP_ARRAY_MAX]
- Parametros med

5.2.1 Detailed Description

Estrutura armazenada na memória com histórico de medições e equipamentos.

Essa estrutura usa uma entrada da estrutura Parametros para armazenar os dados elétricos lidos em associação a um timestamp e um vetor de equipamentos elétricos ligados na rede.

5.2.2 Field Documentation

5.2.2.1 equipamentos

```
uint16_t equipamentos[EQUIP_ARRAY_MAX]
```

Vetor que indica os equipamentos ligados na rede. O índice do vetor indica uma ID de equipamento e o conteúdo indica quantos equipamentos deste tipo estão ligados na rede no momento.

5.2.2.2 med

```
Parametros med
```

Entrada para estrutura Parametros que armazena parâmetros elétricos de interesse.

5.2.2.3 timestamp

```
uint32_t timestamp
```

Marcação de tempo (timestamp) associado a uma medição da rede.

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

- Inc/equipamentos.h

5.3 Parametros Struct Reference

Estrutura com parametros elétricos.

`#include <equipamentos.h>`

Data Fields

- float32_t pot_ap
- float32_t pot_at
- float32_t pot_re
- float32_t harmonicos_RMS [MAX_HARMONICA]
- float32_t thd
- float32_t pf
- float32_t i_rms
- float32_t v_rms

5.3.1 Detailed Description

Estrutura com parametros elétricos.

Esta estrutura é usada para armazenar os parâmetros elétricos de interesse no projeto. Uma entrada desse tipo é usada nas outras estruturas.

5.3.2 Field Documentation

5.3.2.1 harmonicos_RMS

```
float32_t harmonicos_RMS[MAX_HARMONICA]
```

Vetor com a potência das harmônicas.

5.3.2.2 i_rms

```
float32_t i_rms
```

Corrente RMS em Ampères.

5.3.2.3 pf

```
float32_t pf
```

Fator de potência.

5.3.2.4 pot_ap

```
float32_t pot_ap
```

Potencia aparente em Watts.

5.3.2.5 pot_at

```
float32_t pot_at
```

Potencia ativa em Watts.

5.3.2.6 pot_re

```
float32_t pot_re
```

Potencia reativa em Watts.

5.3.2.7 thd

```
float32_t thd
```

Distorção harmônica total.

5.3.2.8 v_rms

```
float32_t v_rms
```

Tensão RMS em Volts.

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

- Inc/equipamentos.h

6 File Documentation

6.1 Inc/adc.h File Reference

Header com funções de configuração dos ADCs.

```
#include "stm32f4xx_hal.h"
#include "main.h"
```

Functions

- void Error_Handler (void)

  This function is executed in case of error occurrence.
- void MX_ADC1_Init (void)

  Inicialização do ADC 1.
- void MX_ADC2_Init (void)

  Inicialização do ADC 2.

Variables

- ADC_HandleTypeDef hadc1

  Handler para estrutura do ADC 1.
- ADC_HandleTypeDef hadc2

  Handler para estrutura do ADC 2.

6.1.1 Detailed Description

Header com funções de configuração dos ADCs.

Author

ST

6.1.2 Function Documentation

6.1.2.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Parameters

None	

Return values

None	

6.1.2.2 MX_ADC1_Init()

```
void MX_ADC1_Init (
             void  )
```

Inicialização do ADC 1.

Return values

None	

Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of conversion)

Configure the ADC multi-mode

Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.

6.1.2.3 MX_ADC2_Init()

```
void MX_ADC2_Init (
             void  )
```

Inicialização do ADC 2.

Return values

None	

Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of conversion)

Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.

6.1.3 Variable Documentation

6.1.3.1 hadc1

```
ADC_HandleTypeDef hadc1
```

Handler para estrutura do ADC 1.

6.1.3.2 hadc2

```
ADC_HandleTypeDef hadc2
```

Handler para estrutura do ADC 2.

6.2 Inc/adc_util.h File Reference

Implementação das funções de auxílio no uso dos ADCs.

```
#include "main.h"
#include "adc.h"
#include "arm_math.h"
```

Functions

- void ADCConvertBuffer (uint32_t *entrada, float32_t *saida, uint32_t n, float32_t a, float32_t b)

  *Converte um buffer de entrada uint32_t em float32_t usando a transformação linear y = a*x + b.*

6.2.1 Detailed Description

Implementação das funções de auxílio no uso dos ADCs.

Author

Gustavo

6.2.2 Function Documentation

6.2.2.1 ADCConvertBuffer()

```
void ADCConvertBuffer (
            uint32_t * entrada,
            float32_t * saida,
            uint32_t n,
            float32_t a,
            float32_t b )
```

Converte um buffer de entrada uint32_t em float32_t usando a transformação linear y = a∗x + b.

Parameters

entrada	buffer de inteiros com valores entre 0 e 4095 (leitura do ADC)
saida	buffer de saída float
a	termo linear da equação y = ax + b
b	termo constante da equação y = ax + b

Return values

None	

6.3 Inc/calculos_eletricos.h File Reference

Implementação das funções que realizam os cálculos dos parâmetros elétricos.

Functions

- void ∗ retornaSIN (float32_t ∗array, uint32_t f, float32_t fs, uint32_t n, float32_t phase)

  DEPRECATED Cria um vetor com sinal senoidal para testes.
- float32_t retornaRMS (float32_t g, float32_t ∗array, uint32_t size, uint32_t start)

  Retorna o valor RMS de um vetor.
- float32_t retornaPOTATIVA (float32_t gv, float32_t gi, float32_t ∗array_tensao, float32_t ∗array_corrente, uint32_t size, uint32_t start)

  Retorna a potência ativa resultante entre um vetor de tensão e um vetor de corrente.

- float32_t retornaPOTAPARENTE (float32_t gv, float32_t gi, float32_t vrms, float32_t irms)

  Retorna a potência aparente resultante a partir dos valores RMS de tensão e corrente já calculados.
- float32_t retornaPOTREATIVA (float32_t potaparente, float32_t potativa)

  Retorna a potência reativa resultante a partir dos valores de potência ativa e aparente já calculados.
- float32_t retornaFP (float32_t potaparente, float32_t potativa)

  Retorna o fator de potência a partir dos valores de potência ativa e aparente já calculados.
- void retornaRMSHARMONICOS (float32_t *i_rms_harmonicos, float32_t *array_corrente, uint32_t size, uint32_t h, float32_t g, uint32_t n, uint32_t start)

  Retorna a potência das harmônicas de um vetor com dados de corrente.
- void retornaMEDIACICLOS (float32_t *array_in, float32_t *array_out, uint32_t size, uint32_t n, uint32_t start)

  Tira a média de n ciclos para cálculo de harmônicas.
- float32_t retornaTHD (float32_t *array_in)

  Retorna o THD a partir de um vetor de harmônicas.

6.3.1 Detailed Description

Implementação das funções que realizam os cálculos dos parâmetros elétricos.

Author

André e Gustavo

6.3.2 Function Documentation

6.3.2.1 retornaFP()

```
float32_t retornaFP (
            float32_t potaparente,
            float32_t potativa )
```

Retorna o fator de potência a partir dos valores de potência ativa e aparente já calculados.

Parameters

potaparente	potência aparente.
potativa	potência ativa.

Return values

fp	fator de reativa calculada.

6.3.2.2 retornaMEDIACICLOS()

```
void retornaMEDIACICLOS (
            float32_t * array_in,
```

```
            float32_t * array_out,
            uint32_t size,
            uint32_t n,
            uint32_t start )
```

Tira a média de n ciclos para cálculo de harmônicas.

Parameters

array_in	ponteiro com dados de entrada. Tamanho size.
array_out	ponteiro com dados de saída. Tamanho (size-start)/n.
size	número de elementos no vetor de dados.
n	número de ciclos de 60 Hz no vetor de entrada.
start	posição inicial do vetor de dados. Usado para que cálculo ocorra sobre um número inteiro de períodos de 60 Hz.

Return values

None	

6.3.2.3 retornaPOTAPARENTE()

```
float32_t retornaPOTAPARENTE (
            float32_t gv,
            float32_t gi,
            float32_t vrms,
            float32_t irms )
```

Retorna a potência aparente resultante a partir dos valores RMS de tensão e corrente já calculados.

Parameters

gi	ganho para ajuste do valor de corrente (não utilizado).
gv	ganho para ajuste do valor de tensão (não utilizado).
vrms	tensão RMS.
irms	corrente RMS.

Return values

potaparente	potência aparente calculada.

6.3.2.4 retornaPOTATIVA()

```
float32_t retornaPOTATIVA (
            float32_t gv,
            float32_t gi,
```

```
            float32_t * array_tensao,
            float32_t * array_corrente,
            uint32_t size,
            uint32_t start )
```

Retorna a potência ativa resultante entre um vetor de tensão e um vetor de corrente.

Parameters

gi	ganho para ajuste do valor de corrente (não utilizado).
gv	ganho para ajuste do valor de tensão (não utilizado).
array_tensao	ponteiro para o vetor de dados de tensão.
array_corrente	ponteiro para o vetor de dados de corrente.
size	número de elementos no vetor de dados.
start	posição inicial do vetor de dados. Usado para que cálculo ocorra sobre um número inteiro de períodos de 60 Hz.

Return values

potativa	potência ativa calculada.

6.3.2.5 retornaPOTREATIVA()

```
float32_t retornaPOTREATIVA (
            float32_t potaparente,
            float32_t potativa )
```

Retorna a potência reativa resultante a partir dos valores de potência ativa e aparente já calculados.

Parameters

potaparente	potência aparente.
potativa	potência ativa.

Return values

potreativa	potência reativa calculada.

6.3.2.6 retornaRMS()

```
float32_t retornaRMS (
            float32_t g,
            float32_t * array,
            uint32_t size,
            uint32_t start )
```

Retorna o valor RMS de um vetor.

Parameters

g	ganho para ajuste do valor (não utilizado).
array	ponteiro para o vetor de dados.
size	número de elementos no vetor de dados.
start	posição inicial do vetor de dados. Usado para que cálculo ocorra sobre um número inteiro de períodos de 60 Hz.

Return values

arrayrms	valor RMS do vetor.

6.3.2.7 retornaRMSHARMONICOS()

```
void retornaRMSHARMONICOS (
            float32_t * i_rms_harmonicos,
            float32_t * array_corrente,
            uint32_t size,
            uint32_t h,
            float32_t g,
            uint32_t n,
            uint32_t start )
```

Retorna a potência das harmônicas de um vetor com dados de corrente.

Parameters

i_rms_harmonicos	ponteiro para vetor de saída com potência das harmônicas.
array_corrente	ponteiro de entrada com valores dos dados de corrente.
g	ganho para ajuste do valor (não utilizado).
size	número de elementos no vetor de dados.
h	número de elementos no vetor de harmônicas.
n	número de ciclos de 60 Hz no vetor de entrada.
start	posição inicial do vetor de dados. Usado para que cálculo ocorra sobre um número inteiro de períodos de 60 Hz.

Return values

None	

6.3.2.8 retornaSIN()

```
void* retornaSIN (
            float32_t * array,
            uint32_t f,
            float32_t fs,
```

```
            uint32_t n,
            float32_t phase )
```

DEPRECATED Cria um vetor com sinal senoidal para testes.

Parameters

array	vetor seno
f	frequencia
fs	frequencia de amostragem
n	numero de pontos por periodo
phase	fase do seno

Return values

None	

6.3.2.9 retornaTHD()

```
float32_t retornaTHD (
            float32_t * array_in )
```

Retorna o THD a partir de um vetor de harmônicas.

Parameters

array_in	ponteiro para vetor de harmônicas.

Return values

thd	Distorção Harmônica Total calculadas.

6.4 Inc/defines.h File Reference

Definições dos parâmetros do projeto.

Macros

- #define MEM_SIZE (uint32_t) 4

  Tamanho da memória de histórico de medições.
- #define BUFFER_SIZE (uint32_t) 2048

  Tamanho do buffer de aquisição do ADC.
- #define DIZIMACAO (uint32_t) 4

  Fator de dizimação.

- #define FILTER_TAP_NUM (uint32_t) 29

  Número de taps do filtro FIR.
- #define BUFFER_DIZ (uint32_t) (BUFFER_SIZE/DIZIMACAO)

  Tamanho do buffer dizimado.
- #define MS2H (float32_t) 1/3600000

  Fator de conversão de milisegundo para hora.
- #define F_SAMP (uint32_t) 12000

  Frequência de amostragem do sistema.
- #define F_SAMP_DIZ (uint32_t) (F_SAMP/DIZIMACAO)

  Frequência de amostragem após dizimação.
- #define PPP (uint32_t) (F_SAMP_DIZ / 60)

  Pontos por período.
- #define N_START (uint32_t) (BUFFER_DIZ-(BUFFER_DIZ/PPP)∗PPP)

  Número de pontos no buffer dizimado que não preenchem um período de 60 Hz.
- #define N_PERIODOS (uint32_t) ((BUFFER_DIZ-N_START)/PPP)

  Número inteiro de períodos de 60 Hz no buffer dizimado.
- #define TENSAO_A (float32_t) -0.0992

  Fator de conversão A da calibração de tensão.
- #define TENSAO_B (float32_t) 200.8922

  Fator de conversão B da calibração de tensão.
- #define CORRENTE_A (float32_t) 3.3/4095∗10

  Fator de conversão A da calibração de corrente.
- #define CORRENTE_B (float32_t) -2048∗CORRENTE_A

  Fator de conversão B da calibração de corrente.
- #define RMS_TOLERANCIA (float32_t) 7

  Tolerância do evento de corrente, em %.
- #define RMS_UPPERBOUND (float32_t) (100+RMS_TOLERANCIA)/100

  Limite superior da tolerância do evento de corrente.
- #define RMS_LOWERBOUND (float32_t) (100-RMS_TOLERANCIA)/100

  Limite inferior da tolerância do evento de corrente.
- #define GV 1

  DEPRECATED Ganho de corrente para ajuste.
- #define GI 1

  DEPRECATED Ganho de tensão para ajuste.
- #define MAX_HARMONICA 10

  Número de harmônicas consideradas.
- #define EQUIP_ARRAY_MAX 16

  Número máximo de equipamentos na base de dados.

6.4.1 Detailed Description

Definições dos parâmetros do projeto.

Author

Gustavo

6.4.2 Macro Definition Documentation

6.4.2.1 BUFFER_DIZ

```
#define BUFFER_DIZ (uint32_t) (BUFFER_SIZE/DIZIMACAO)
```

Tamanho do buffer dizimado.

6.4.2.2 BUFFER_SIZE

```
#define BUFFER_SIZE (uint32_t) 2048
```

Tamanho do buffer de aquisição do ADC.

6.4.2.3 CORRENTE_A

```
#define CORRENTE_A (float32_t) 3.3/4095*10
```

Fator de conversão A da calibração de corrente.

6.4.2.4 CORRENTE_B

```
#define CORRENTE_B (float32_t) -2048*CORRENTE_A
```

Fator de conversão B da calibração de corrente.

6.4.2.5 DIZIMACAO

```
#define DIZIMACAO (uint32_t) 4
```

Fator de dizimação.

6.4.2.6 EQUIP_ARRAY_MAX

```
#define EQUIP_ARRAY_MAX 16
```

Número máximo de equipamentos na base de dados.

6.4.2.7 F_SAMP

```
#define F_SAMP (uint32_t) 12000
```

Frequência de amostragem do sistema.

6.4.2.8 F_SAMP_DIZ

```
#define F_SAMP_DIZ (uint32_t) (F_SAMP/DIZIMACAO)
```

Frequência de amostragem após dizimação.

6.4.2.9 FILTER_TAP_NUM

```
#define FILTER_TAP_NUM (uint32_t) 29
```

Número de taps do filtro FIR.

6.4.2.10 GI

```
#define GI 1
```

DEPRECATED Ganho de tensão para ajuste.

6.4.2.11 GV

```
#define GV 1
```

DEPRECATED Ganho de corrente para ajuste.

6.4.2.12 MAX_HARMONICA

```
#define MAX_HARMONICA 10
```

Número de harmônicas consideradas.

6.4.2.13 MEM_SIZE

```
#define MEM_SIZE (uint32_t) 4
```

Tamanho da memória de histórico de medições.

6.4.2.14 MS2H

```
#define MS2H (float32_t) 1/3600000
```

Fator de conversão de milisegundo para hora.

6.4.2.15 N_PERIODOS

```
#define N_PERIODOS (uint32_t) ((BUFFER_DIZ-N_START)/PPP)
```

Número inteiro de períodos de 60 Hz no buffer dizimado.

6.4.2.16 N_START

```
#define N_START (uint32_t) (BUFFER_DIZ-(BUFFER_DIZ/PPP)*PPP)
```

Número de pontos no buffer dizimado que não preenchem um período de 60 Hz.

6.4.2.17 PPP

```
#define PPP (uint32_t) (F_SAMP_DIZ / 60)
```

Pontos por período.

6.4.2.18 RMS_LOWERBOUND

```
#define RMS_LOWERBOUND (float32_t) (100-RMS_TOLERANCIA)/100
```

Limite inferior da tolerância do evento de corrente.

6.4.2.19 RMS_TOLERANCIA

```
#define RMS_TOLERANCIA (float32_t) 7
```

Tolerância do evento de corrente, em %.

6.4.2.20 RMS_UPPERBOUND

```
#define RMS_UPPERBOUND (float32_t) (100+RMS_TOLERANCIA)/100
```

Limite superior da tolerância do evento de corrente.

6.4.2.21 TENSAO_A

```
#define TENSAO_A (float32_t) -0.0992
```

Fator de conversão A da calibração de tensão.

6.4.2.22 TENSAO_B

```
#define TENSAO_B (float32_t) 200.8922
```

Fator de conversão B da calibração de tensão.

6.5 Inc/dizimacao.h File Reference

Código para dizimação: Filtro FIR e downsampling usando biblioteca CMSIS.

```
#include "defines.h"
#include "arm_math.h"
```

Functions

- void initializeFIR (arm_fir_decimate_instance_f32 *S)

  Inicializa estrutura da biblioteca CMSIS usada para dizimação.

Variables

- static float pCoeffs [FILTER_TAP_NUM]

  Coeficientes do filtro FIR.
- float32_t pState [FILTER_TAP_NUM+BUFFER_SIZE-1]

  Buffer auxiliar para estrutura da biblioteca CMSIS.
- arm_fir_decimate_instance_f32 S = {DIZIMACAO, FILTER_TAP_NUM, pCoeffs, pState}

  Estrutura da biblioteca CMSIS usada para dizimação.

6.5.1 Detailed Description

Código para dizimação: Filtro FIR e downsampling usando biblioteca CMSIS.

Author

Bruno e Gustavo

6.5.2 Function Documentation

6.5.2.1 initializeFIR()

```
void initializeFIR (
            arm_fir_decimate_instance_f32 * S )
```

Inicializa estrutura da biblioteca CMSIS usada para dizimação.

Parameters

S	estrutura da biblioteca CMSIS usada para dizimação.

Return values

None	

6.5.3 Variable Documentation

6.5.3.1 pCoeffs

```
float pCoeffs[FILTER_TAP_NUM]  [static]
```

Coeficientes do filtro FIR.

6.5.3.2 pState

```
float32_t pState[FILTER_TAP_NUM+BUFFER_SIZE-1]
```

Buffer auxiliar para estrutura da biblioteca CMSIS.

6.5.3.3 S

```
arm_fir_decimate_instance_f32 S = {DIZIMACAO, FILTER_TAP_NUM, pCoeffs, pState}
```

Estrutura da biblioteca CMSIS usada para dizimação.

6.6 Inc/dma.h File Reference

Header das funções de configuração das DMAs.

```
#include "stm32f4xx_hal.h"
#include "main.h"
```

Functions

- void Error_Handler (void)

  This function is executed in case of error occurrence.
- void MX_DMA_Init (void)

  Inicialização dos canais do DMA 2.

Variables

- DMA_HandleTypeDef hdma_memtomem_dma2_stream1

  Handler para estrutura do stream 1 do DMA 2 para transferência MemToMem.
- DMA_HandleTypeDef hdma_memtomem_dma2_stream3

  Handler para estrutura do stream 3 do DMA 2 para transferência MemToMem.

6.6.1 Detailed Description

Header das funções de configuração das DMAs.

Author

ST

6.6.2 Function Documentation

6.6.2.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Parameters

None	

Return values

None	

6.6.2.2 MX_DMA_Init()

```
void MX_DMA_Init (
             void  )
```

Inicialização dos canais do DMA 2.

Return values

None	Inicialização dos canais do DMA 2.

Enable DMA controller clock Configure DMA for memory to memory transfers hdma_memtomem_dma2_stream1 hdma_memtomem_dma2_stream3

Return values

None	

6.6.3 Variable Documentation

6.6.3.1 hdma_memtomem_dma2_stream1

```
DMA_HandleTypeDef hdma_memtomem_dma2_stream1
```

Handler para estrutura do stream 1 do DMA 2 para transferência MemToMem.

6.6.3.2 hdma_memtomem_dma2_stream3

```
DMA_HandleTypeDef hdma_memtomem_dma2_stream3
```

Handler para estrutura do stream 3 do DMA 2 para transferência MemToMem.

6.7 Inc/equipamentos.h File Reference

Header com as estruturas do projeto e funções que manipulam essas estruturas.

```
#include "arm_math.h"
#include "defines.h"
#include "calculos_eletricos.h"
#include <math.h>
```

Data Structures

- struct Parametros

  Estrutura com parametros elétricos.
- struct Medicao

  Estrutura armazenada na memória com histórico de medições e equipamentos.
- struct Equipamento

  Estrutura da base de dados de equipamentos.

Typedefs

- typedef struct Parametros Parametros

  Estrutura com parametros elétricos.
- typedef struct Medicao Medicao

  Estrutura armazenada na memória com histórico de medições e equipamentos.

Functions

- void CadastroDeEquipamento (Equipamento *, uint16_t, char nome[40], float32_t, float32_t, float32_t, float32_t, float32_t *)

  Cria estrutura com informações de um novo equipamento.
- void DeltaParam (Parametros *, Parametros *, Parametros *, char *)

  Calcula delta entre duas estruturas Parametros .
- float ComparacaoDeEquipamentos (Equipamento *, Parametros *)

  DEPRECATED Calcula delta entre duas estruturas Parametros .
- void InitMedicao (Medicao *)

  Inicializa uma estrutura Medicao .
- void InitBaseDeDados (Equipamento *)

  Inicializa uma vetor (base de dados) de estruturas Equipamento .
- int IdentificarEquipamento (Equipamento *, Parametros *)

  Função com implementação básica de um algoritmo de identificação de equipamento na rede.

6.7.1 Detailed Description

Header com as estruturas do projeto e funções que manipulam essas estruturas.

Author

Bruno e Gustavo

6.7.2 Typedef Documentation

6.7.2.1 Medicao

```
typedef struct Medicao Medicao
```

Estrutura armazenada na memória com histórico de medições e equipamentos.

Essa estrutura usa uma entrada da estrutura Parametros para armazenar os dados elétricos lidos em associação a um timestamp e um vetor de equipamentos elétricos ligados na rede.

6.7.2.2 Parametros

```
typedef struct Parametros Parametros
```

Estrutura com parametros elétricos.

Esta estrutura é usada para armazenar os parâmetros elétricos de interesse no projeto. Uma entrada desse tipo é usada nas outras estruturas.

6.7.3 Function Documentation

6.7.3.1 CadastroDeEquipamento()

```
void CadastroDeEquipamento (
            Equipamento * equip,
            uint16_t ID,
            char nome[40],
            float32_t pot_at,
            float32_t pot_re,
            float32_t thd,
            float32_t pf,
            float32_t * harmonicos )
```

Cria estrutura com informações de um novo equipamento.

Parameters

equip	ponteiro para estrutura Equipamento .
nome	char∗ com nome do equipamento.
pot_at	potencia ativa.
pot_re	potencia reativa.
thd	taxa de distorção harmonica.
pf	fator de potencia.
harmonicos	vetor de harmonicas.

Return values

None	

6.7.3.2 ComparacaoDeEquipamentos()

```
float ComparacaoDeEquipamentos (
            Equipamento * equip1,
            Parametros * medida )
```

DEPRECATED Calcula delta entre duas estruturas Parametros .

Return values

None	

6.7.3.3 DeltaParam()

```
void DeltaParam (
            Parametros * medida_nova,
            Parametros * medida_velha,
            Parametros * delta,
            char * new )
```

Calcula delta entre duas estruturas Parametros .

Parameters

medida_nova	ponteiro para uma estrutura Parametros que contém os parâmetros da medição mais recente.
medida_velha	ponteiro para uma estrutura Parametros que contém os parâmetros da medição anterior.
delta	ponteiro para uma estrutura Parametros que armazana o delta entre as duas entradas.
new	ponteiro para um char que indicará se um equipamento foi adicionado ou retirado da rede.

Return values

None	

6.7.3.4 IdentificarEquipamento()

```
int IdentificarEquipamento (
            Equipamento * BaseDados,
            Parametros * Medicao )
```

Função com implementação básica de um algoritmo de identificação de equipamento na rede.

Parameters

BaseDados	ponteiro para um vetor de Equipamento sobre o qual é feita a busca da Medicao.
Medicao	ponteiro para Parametros que contém a medição do delta na rede. Representa um equipamento acrescido ou retirado da rede.

Return values

min	Índice da base de dados que contém o equipamento identificado.

6.7.3.5 InitBaseDeDados()

```
void InitBaseDeDados (
            Equipamento * BaseDados )
```

Inicializa uma vetor (base de dados) de estruturas Equipamento .

Parameters

BaseDados	ponteiro para um vetor de Equipamento que será inicializada com zeros.

Return values

None	

6.7.3.6 InitMedicao()

```
void InitMedicao (
            Medicao * medicao )
```

Inicializa uma estrutura Medicao .

Parameters

medicao	ponteiro para uma estrutura Medicao que será inicializada com zeros.

Return values

None	

6.8 Inc/gpio.h File Reference

Header das funções de configuração do GPIO.

```
#include "stm32f4xx_hal.h"
#include "main.h"
```

Functions

- void MX_GPIO_Init (void)
  Inicialização do GPIO.

6.8.1 Detailed Description

Header das funções de configuração do GPIO.

Author

ST

6.8.2 Function Documentation

6.8.2.1 MX_GPIO_Init()

```
void MX_GPIO_Init (
            void  )
```

Inicialização do GPIO.

Return values

None	

6.9 Inc/main.h File Reference

Header do main. Não utilizado.

6.9.1 Detailed Description

Header do main. Não utilizado.

Author

ST

6.10 Inc/stm32f4xx_hal_conf.h File Reference

HAL configuration file.

```
#include "main.h"
#include "stm32f4xx_hal_rcc.h"
#include "stm32f4xx_hal_gpio.h"
#include "stm32f4xx_hal_dma.h"
#include "stm32f4xx_hal_cortex.h"
#include "stm32f4xx_hal_adc.h"
#include "stm32f4xx_hal_flash.h"
#include "stm32f4xx_hal_pwr.h"
#include "stm32f4xx_hal_tim.h"
#include "stm32f4xx_hal_uart.h"
```

Macros

- #define HAL_MODULE_ENABLED

  This is the list of modules to be used in the HAL driver.
- #define HAL_ADC_MODULE_ENABLED
- #define HAL_TIM_MODULE_ENABLED
- #define HAL_UART_MODULE_ENABLED
- #define HAL_GPIO_MODULE_ENABLED
- #define HAL_DMA_MODULE_ENABLED
- #define HAL_RCC_MODULE_ENABLED
- #define HAL_FLASH_MODULE_ENABLED
- #define HAL_PWR_MODULE_ENABLED
- #define HAL_CORTEX_MODULE_ENABLED
- #define HSE_VALUE ((uint32_t)8000000U)

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

- #define HSI_VALUE ((uint32_t)16000000U)

  Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).
- #define LSI_VALUE ((uint32_t)32000U)

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

  External Low Speed oscillator (LSE) value.
- #define LSE_STARTUP_TIMEOUT ((uint32_t)5000U)
- #define EXTERNAL_CLOCK_VALUE ((uint32_t)12288000U)

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

  This is the HAL system configuration section.
- #define TICK_INT_PRIORITY ((uint32_t)0U)
- #define USE_RTOS 0U
- #define PREFETCH_ENABLE 1U
- #define INSTRUCTION_CACHE_ENABLE 1U
- #define DATA_CACHE_ENABLE 1U
- #define MAC_ADDR0 2U

  Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.
- #define MAC_ADDR1 0U
- #define MAC_ADDR2 0U
- #define MAC_ADDR3 0U
- #define MAC_ADDR4 0U
- #define MAC_ADDR5 0U
- #define ETH_RX_BUF_SIZE ETH_MAX_PACKET_SIZE /∗ buffer size for receive ∗/
- #define ETH_TX_BUF_SIZE ETH_MAX_PACKET_SIZE /∗ buffer size for transmit ∗/
- #define ETH_RXBUFNB ((uint32_t)4U) /∗ 4 Rx buffers of size ETH_RX_BUF_SIZE ∗/
- #define ETH_TXBUFNB ((uint32_t)4U) /∗ 4 Tx buffers of size ETH_TX_BUF_SIZE ∗/
- #define DP83848_PHY_ADDRESS 0x01U
- #define PHY_RESET_DELAY ((uint32_t)0x000000FFU)
- #define PHY_CONFIG_DELAY ((uint32_t)0x00000FFFU)
- #define PHY_READ_TO ((uint32_t)0x0000FFFFU)
- #define PHY_WRITE_TO ((uint32_t)0x0000FFFFU)
- #define PHY_BCR ((uint16_t)0x0000U)
- #define PHY_BSR ((uint16_t)0x0001U)
- #define PHY_RESET ((uint16_t)0x8000U)
- #define PHY_LOOPBACK ((uint16_t)0x4000U)
- #define PHY_FULLDUPLEX_100M ((uint16_t)0x2100U)
- #define PHY_HALFDUPLEX_100M ((uint16_t)0x2000U)
- #define PHY_FULLDUPLEX_10M ((uint16_t)0x0100U)
- #define PHY_HALFDUPLEX_10M ((uint16_t)0x0000U)
- #define PHY_AUTONEGOTIATION ((uint16_t)0x1000U)
- #define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200U)
- #define PHY_POWERDOWN ((uint16_t)0x0800U)
- #define PHY_ISOLATE ((uint16_t)0x0400U)
- #define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020U)
- #define PHY_LINKED_STATUS ((uint16_t)0x0004U)
- #define PHY_JABBER_DETECTION ((uint16_t)0x0002U)
- #define PHY_SR ((uint16_t)0x10U)
- #define PHY_SPEED_STATUS ((uint16_t)0x0002U)
- #define PHY_DUPLEX_STATUS ((uint16_t)0x0004U)
- #define USE_SPI_CRC 0U
- #define assert_param(expr) ((void)0U)

  Include module's header file.

6.10.1 Detailed Description

HAL configuration file.

Attention

6.10.2 Macro Definition Documentation

6.10.2.1 assert_param

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

Include module's header file.

6.10.2.2 DATA_CACHE_ENABLE

```
#define DATA_CACHE_ENABLE 1U
```

6.10.2.3 DP83848_PHY_ADDRESS

```
#define DP83848_PHY_ADDRESS 0x01U
```

6.10.2.4 ETH_RX_BUF_SIZE

```
#define ETH_RX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for receive */
```

6.10.2.5 ETH_RXBUFNB

```
#define ETH_RXBUFNB ((uint32_t)4U) /* 4 Rx buffers of size ETH_RX_BUF_SIZE */
```

6.10.2.6 ETH_TX_BUF_SIZE

```
#define ETH_TX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for transmit */
```

6.10.2.7 ETH_TXBUFNB

```
#define ETH_TXBUFNB ((uint32_t)4U) /* 4 Tx buffers of size ETH_TX_BUF_SIZE */
```

6.10.2.8 EXTERNAL_CLOCK_VALUE

```
#define EXTERNAL_CLOCK_VALUE ((uint32_t)12288000U)
```

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

Value of the External audio frequency in Hz

6.10.2.9 HAL_ADC_MODULE_ENABLED

```
#define HAL_ADC_MODULE_ENABLED
```

6.10.2.10 HAL_CORTEX_MODULE_ENABLED

```
#define HAL_CORTEX_MODULE_ENABLED
```

6.10.2.11 HAL_DMA_MODULE_ENABLED

```
#define HAL_DMA_MODULE_ENABLED
```

6.10.2.12 HAL_FLASH_MODULE_ENABLED

```
#define HAL_FLASH_MODULE_ENABLED
```

6.10.2.13 HAL_GPIO_MODULE_ENABLED

```
#define HAL_GPIO_MODULE_ENABLED
```

6.10.2.14 HAL_MODULE_ENABLED

```
#define HAL_MODULE_ENABLED
```

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

6.10.2.15 HAL_PWR_MODULE_ENABLED

```
#define HAL_PWR_MODULE_ENABLED
```

6.10.2.16 HAL_RCC_MODULE_ENABLED

```
#define HAL_RCC_MODULE_ENABLED
```

6.10.2.17 HAL_TIM_MODULE_ENABLED

```
#define HAL_TIM_MODULE_ENABLED
```

6.10.2.18 HAL_UART_MODULE_ENABLED

```
#define HAL_UART_MODULE_ENABLED
```

6.10.2.19 HSE_STARTUP_TIMEOUT

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

Time out for HSE start up, in ms

6.10.2.20 HSE_VALUE

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

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

Value of the External oscillator in Hz

6.10.2.21 HSI_VALUE

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

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

Value of the Internal oscillator in Hz

6.10.2.22 INSTRUCTION_CACHE_ENABLE

```
#define INSTRUCTION_CACHE_ENABLE 1U
```

6.10.2.23 LSE_STARTUP_TIMEOUT

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

Time out for LSE start up, in ms

6.10.2.24 LSE_VALUE

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

External Low Speed oscillator (LSE) value.

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

6.10.2.25 LSI_VALUE

```
#define LSI_VALUE ((uint32_t)32000U)
```

Internal Low Speed oscillator (LSI) value.

LSI Typical Value in Hz

6.10.2.26 MAC_ADDR0

```
#define MAC_ADDR0 2U
```

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

6.10.2.27 MAC_ADDR1

```
#define MAC_ADDR1 0U
```

6.10.2.28 MAC_ADDR2

```
#define MAC_ADDR2 0U
```

6.10.2.29 MAC_ADDR3

```
#define MAC_ADDR3 0U
```

6.10.2.30 MAC_ADDR4

```
#define MAC_ADDR4 0U
```

6.10.2.31 MAC_ADDR5

```
#define MAC_ADDR5 0U
```

6.10.2.32 PHY_AUTONEGO_COMPLETE

```
#define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020U)
```

Auto-Negotiation process completed

6.10.2.33 PHY_AUTONEGOTIATION

```
#define PHY_AUTONEGOTIATION ((uint16_t)0x1000U)
```

Enable auto-negotiation function

6.10.2.34 PHY_BCR

```
#define PHY_BCR ((uint16_t)0x0000U)
```

Transceiver Basic Control Register

6.10.2.35 PHY_BSR

```
#define PHY_BSR ((uint16_t)0x0001U)
```

Transceiver Basic Status Register

6.10.2.36 PHY_CONFIG_DELAY

```
#define PHY_CONFIG_DELAY ((uint32_t)0x00000FFFU)
```

6.10.2.37 PHY_DUPLEX_STATUS

```
#define PHY_DUPLEX_STATUS ((uint16_t)0x0004U)
```

PHY Duplex mask

6.10.2.38 PHY_FULLDUPLEX_100M

```
#define PHY_FULLDUPLEX_100M ((uint16_t)0x2100U)
```

Set the full-duplex mode at 100 Mb/s

6.10.2.39 PHY_FULLDUPLEX_10M

```
#define PHY_FULLDUPLEX_10M ((uint16_t)0x0100U)
```

Set the full-duplex mode at 10 Mb/s

6.10.2.40 PHY_HALFDUPLEX_100M

```
#define PHY_HALFDUPLEX_100M ((uint16_t)0x2000U)
```

Set the half-duplex mode at 100 Mb/s

6.10.2.41 PHY_HALFDUPLEX_10M

```
#define PHY_HALFDUPLEX_10M ((uint16_t)0x0000U)
```

Set the half-duplex mode at 10 Mb/s

6.10.2.42 PHY_ISOLATE

```
#define PHY_ISOLATE ((uint16_t)0x0400U)
```

Isolate PHY from MII

6.10.2.43 PHY_JABBER_DETECTION

```
#define PHY_JABBER_DETECTION ((uint16_t)0x0002U)
```

Jabber condition detected

6.10.2.44 PHY_LINKED_STATUS

```
#define PHY_LINKED_STATUS ((uint16_t)0x0004U)
```

Valid link established

6.10.2.45 PHY_LOOPBACK

```
#define PHY_LOOPBACK ((uint16_t)0x4000U)
```

Select loop-back mode

6.10.2.46 PHY_POWERDOWN

```
#define PHY_POWERDOWN ((uint16_t)0x0800U)
```

Select the power down mode

6.10.2.47 PHY_READ_TO

```
#define PHY_READ_TO ((uint32_t)0x0000FFFFU)
```

6.10.2.48 PHY_RESET

```
#define PHY_RESET ((uint16_t)0x8000U)
```

PHY Reset

6.10.2.49 PHY_RESET_DELAY

```
#define PHY_RESET_DELAY ((uint32_t)0x000000FFU)
```

6.10.2.50 PHY_RESTART_AUTONEGOTIATION

```
#define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200U)
```

Restart auto-negotiation function

6.10.2.51 PHY_SPEED_STATUS

```
#define PHY_SPEED_STATUS ((uint16_t)0x0002U)
```

PHY Speed mask

6.10.2.52 PHY_SR

```
#define PHY_SR ((uint16_t)0x10U)
```

PHY status register Offset

6.10.2.53 PHY_WRITE_TO

```
#define PHY_WRITE_TO ((uint32_t)0x0000FFFFU)
```

6.10.2.54 PREFETCH_ENABLE

```
#define PREFETCH_ENABLE 1U
```

6.10.2.55 TICK_INT_PRIORITY

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

tick interrupt priority

6.10.2.56 USE_RTOS

```
#define USE_RTOS 0U
```

6.10.2.57 USE_SPI_CRC

```
#define USE_SPI_CRC 0U
```

6.10.2.58 VDD_VALUE

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

This is the HAL system configuration section.

Value of VDD in mv

6.11 Inc/stm32f4xx_it.h File Reference

This file contains the headers of the interrupt handlers.

Functions

- void SVC_Handler (void)

  This function handles System service call via SWI instruction.
- void PendSV_Handler (void)

  This function handles Pendable request for system service.
- void SysTick_Handler (void)

  This function handles System tick timer.
- void ADC_IRQHandler (void)

  This function handles ADC1, ADC2 and ADC3 interrupts.
- void USART2_IRQHandler (void)

  This function handles USART2 global interrupt.
- void DMA2_Stream0_IRQHandler (void)

  This function handles DMA2 stream0 global interrupt.
- void DMA2_Stream1_IRQHandler (void)

  This function handles DMA2 stream1 global interrupt.
- void DMA2_Stream2_IRQHandler (void)

  This function handles DMA2 stream2 global interrupt.
- void DMA2_Stream3_IRQHandler (void)

  This function handles DMA2 stream3 global interrupt.

6.11.1 Detailed Description

This file contains the headers of the interrupt handlers.

6.11.2 Function Documentation

6.11.2.1 ADC_IRQHandler()

```
void ADC_IRQHandler (
            void  )
```

This function handles ADC1, ADC2 and ADC3 interrupts.

6.11.2.2 DMA2_Stream0_IRQHandler()

```
void DMA2_Stream0_IRQHandler (
            void  )
```

This function handles DMA2 stream0 global interrupt.

6.11.2.3 DMA2_Stream1_IRQHandler()

```
void DMA2_Stream1_IRQHandler (
            void  )
```

This function handles DMA2 stream1 global interrupt.

6.11.2.4 DMA2_Stream2_IRQHandler()

```
void DMA2_Stream2_IRQHandler (
            void  )
```

This function handles DMA2 stream2 global interrupt.

6.11.2.5 DMA2_Stream3_IRQHandler()

```
void DMA2_Stream3_IRQHandler (
            void  )
```

This function handles DMA2 stream3 global interrupt.

6.11.2.6 PendSV_Handler()

```
void PendSV_Handler (
            void  )
```

This function handles Pendable request for system service.

6.11.2.7 SVC_Handler()

```
void SVC_Handler (
            void  )
```

This function handles System service call via SWI instruction.

6.11.2.8 SysTick_Handler()

```
void SysTick_Handler (
            void  )
```

This function handles System tick timer.

6.11.2.9 USART2_IRQHandler()

```
void USART2_IRQHandler (
            void  )
```

This function handles USART2 global interrupt.

6.12 Inc/tim.h File Reference

Header das funções de configuração do temporizador.

```
#include "stm32f4xx_hal.h"
#include "main.h"
```

Functions

- void Error_Handler (void)
  This function is executed in case of error occurrence.
- void MX_TIM8_Init (void)
  Inicialização dos canais do Timer 8.

Variables

- TIM_HandleTypeDef htim8
  Handler para estrutura do Timer 8.

6.12.1 Detailed Description

Header das funções de configuração do temporizador.

Author

ST

6.12.2 Function Documentation

6.12.2.1 Error_Handler()

```
void Error_Handler (
            void  )
```

This function is executed in case of error occurrence.

Parameters

None	

Return values

None	

6.12.2.2 MX_TIM8_Init()

```
void MX_TIM8_Init (
             void  )
```

Inicialização dos canais do Timer 8.

Return values

None	

6.12.3 Variable Documentation

6.12.3.1 htim8

```
TIM_HandleTypeDef htim8
```

Handler para estrutura do Timer 8.

6.13 Inc/usart.h File Reference

Header das funções de configuração da uart.

```
#include "stm32f4xx_hal.h"
#include "main.h"
```

Functions

- void Error_Handler (void)

  This function is executed in case of error occurrence.
- void MX_USART2_UART_Init (void)

  Inicialização dos canais da UART 2.

Variables

- UART_HandleTypeDef huart2

  Handler para estrutura do UART 2.

6.13.1 Detailed Description

Header das funções de configuração da uart.

Author

ST

6.13.2 Function Documentation

6.13.2.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Parameters

None	

Return values

None	

6.13.2.2 MX_USART2_UART_Init()

```
void MX_USART2_UART_Init (
             void  )
```

Inicialização dos canais da UART 2.

Return values

None	

6.13.3 Variable Documentation

6.13.3.1 huart2

```
UART_HandleTypeDef huart2
```

Handler para estrutura do UART 2.

6.14 Inc/usart_util.h File Reference

Biblioteca com funções úteis para utilizar com a USART2.

Functions

- void USART2_Transmit_Char (char)
  Transmite caractere ASCII pela USART 2.
- void USART2_Transmit_String (char ∗)
  Transmite string pela USART 2.
- void USART2_Transmit_UInt (uint32_t)
  Transmite inteiro unsigned pela USART 2.
- void USART2_Transmit_Int (int)
  Transmite inteiro signed pela USART 2.
- char USART2_Receive_Command (int)
  Recebe caracter de comando pela USART2.
- void USAR2_Receive_Interrupt_Enable (void)
  Habilita interrupção.
- void print (const char ∗,...)
  Função semelhante à printf usando USART 2. Funciona apenas para d e c.

6.14.1 Detailed Description

Biblioteca com funções úteis para utilizar com a USART2.

Author

Gustavo

6.14.2 Function Documentation

6.14.2.1 print()

```
void print (
            const char * send,
             ...  )
```

Função semelhante à printf usando USART 2. Funciona apenas para d e c.

Parameters

send	string formatada a ser transferida pela USART 2

Return values

None	

6.14.2.2 USAR2_Receive_Interrupt_Enable()

```
void USAR2_Receive_Interrupt_Enable (
             void  )
```

Habilita interrupção.

Return values

None	

6.14.2.3 USART2_Receive_Command()

```
char USART2_Receive_Command (
             int timeout )
```

Recebe caracter de comando pela USART2.

Parameters

timeout	Timeout em ms.

Return values

None	

6.14.2.4 USART2_Transmit_Char()

```
void USART2_Transmit_Char (
             char send )
```

Transmite caractere ASCII pela USART 2.

Parameters

send	caracter a ser transferido pela USART 2

Return values

None	

6.14.2.5 USART2_Transmit_Int()

```
void USART2_Transmit_Int (
            int send )
```

Transmite inteiro signed pela USART 2.

Parameters

send	inteiro a ser transferido pela USART 2

Return values

None	

6.14.2.6 USART2_Transmit_String()

```
void USART2_Transmit_String (
            char * send )
```

Transmite string pela USART 2.

Parameters

send	string a ser transferida pela USART 2

Return values

None	

6.14.2.7 USART2_Transmit_UInt()

```
void USART2_Transmit_UInt (
            uint32_t send )
```

Transmite inteiro unsigned pela USART 2.

Parameters

send	inteiro a ser transferido pela USART 2

Return values

None	

6.15 Src/adc.c File Reference

Implementação das funções de configuração dos ADCs.

```
#include "adc.h"
#include "gpio.h"
#include "dma.h"
```

Functions

- void MX_ADC1_Init (void)
  Inicialização do ADC 1.
- void MX_ADC2_Init (void)
  Inicialização do ADC 2.
- void HAL_ADC_MspInit (ADC_HandleTypeDef ∗adcHandle)
  Inicialização da DMA de um ADC.
- void HAL_ADC_MspDeInit (ADC_HandleTypeDef ∗adcHandle)
  Desativações da DMA de um ADC.

Variables

- ADC_HandleTypeDef hadc1
  Handler para estrutura do ADC 1.
- ADC_HandleTypeDef hadc2
  Handler para estrutura do ADC 2.
- DMA_HandleTypeDef hdma_adc1
  Handler para estrutura do DMA do ADC 1.
- DMA_HandleTypeDef hdma_adc2
  Handler para estrutura do DMA do ADC 2.

6.15.1 Detailed Description

Implementação das funções de configuração dos ADCs.

Author

ST

6.15.2 Function Documentation

6.15.2.1 HAL_ADC_MspDeInit()

```
void HAL_ADC_MspDeInit (
            ADC_HandleTypeDef * adcHandle )
```

Desativações da DMA de um ADC.

Parameters

adcHandle	Handle do ADC a ter sua DMA desativada.

Return values

None	

ADC1 GPIO Configuration PA6 ——> ADC1_IN6

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

ADC2 GPIO Configuration PA7 ——> ADC2_IN7

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

6.15.2.2 HAL_ADC_MspInit()

```
void HAL_ADC_MspInit (
             ADC_HandleTypeDef * adcHandle )
```

Inicialização da DMA de um ADC.

Parameters

adcHandle	Handle do ADC a ter sua DMA configurada.

Return values

None	

ADC1 GPIO Configuration PA6 ——> ADC1_IN6

ADC2 GPIO Configuration PA7 ----—> ADC2_IN7

6.15.2.3 MX_ADC1_Init()

```
void MX_ADC1_Init (
             void  )
```

Inicialização do ADC 1.

Return values

None	

Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of conversion)

Configure the ADC multi-mode

Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.

6.15.2.4 MX_ADC2_Init()

```
void MX_ADC2_Init (
             void  )
```

Inicialização do ADC 2.

Return values

None	

Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of conversion)

Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.

6.15.3 Variable Documentation

6.15.3.1 hadc1

```
ADC_HandleTypeDef hadc1
```

Handler para estrutura do ADC 1.

6.15.3.2 hadc2

```
ADC_HandleTypeDef hadc2
```

Handler para estrutura do ADC 2.

6.15.3.3 hdma_adc1

```
DMA_HandleTypeDef hdma_adc1
```

Handler para estrutura do DMA do ADC 1.

6.15.3.4 hdma_adc2

```
DMA_HandleTypeDef hdma_adc2
```

Handler para estrutura do DMA do ADC 2.

6.16 Src/adc_util.c File Reference

Implementação das funções de auxílio no uso dos ADCs.

```
#include "adc_util.h"
```

Functions

- void ADCConvertBuffer (uint32_t ∗entrada, float32_t ∗saida, uint32_t n, float32_t a, float32_t b)
  Converte um buffer de entrada uint32_t em float32_t usando a transformação linear y = a∗x + b.

6.16.1 Detailed Description

Implementação das funções de auxílio no uso dos ADCs.

Author

Gustavo

6.16.2 Function Documentation

6.16.2.1 ADCConvertBuffer()

```
void ADCConvertBuffer (
            uint32_t * entrada,
            float32_t * saida,
            uint32_t n,
            float32_t a,
            float32_t b )
```

Converte um buffer de entrada uint32_t em float32_t usando a transformação linear y = a∗x + b.

Parameters

entrada	buffer de inteiros com valores entre 0 e 4095 (leitura do ADC)
saida	buffer de saída float
a	termo linear da equação y = ax + b
b	termo constante da equação y = ax + b

Return values

None	

6.17 Src/calculos_eletricos.c File Reference

Implementação das funções que realizam os cálculos dos parâmetros elétricos.

```
#include "stdio.h"
#include "stdlib.h"
#include "math.h"
#include "arm_math.h"
#include "defines.h"
```

Macros

- #define PI_VALUE (float32_t) 3.14159265358
- #define SQRT2 (float32_t) 1.414213562373095

Functions

- void * retornaSIN (float32_t *array, uint32_t f, float32_t fs, uint32_t n, float32_t phase)

  DEPRECATED Cria um vetor com sinal senoidal para testes.
- float32_t retornaRMS (float32_t g, float32_t *array, uint32_t size, uint32_t start)

  Retorna o valor RMS de um vetor.
- float32_t retornaPOTATIVA (float32_t gv, float32_t gi, float32_t *array_tensao, float32_t *array_corrente, uint32_t size, uint32_t start)

  Retorna a potência ativa resultante entre um vetor de tensão e um vetor de corrente.
- float32_t retornaPOTAPARENTE (float32_t gv, float32_t gi, float32_t vrms, float32_t irms)

  Retorna a potência aparente resultante a partir dos valores RMS de tensão e corrente já calculados.
- float32_t retornaPOTREATIVA (float32_t potaparente, float32_t potativa)

  Retorna a potência reativa resultante a partir dos valores de potência ativa e aparente já calculados.
- float32_t retornaFP (float32_t potaparente, float32_t potativa)

  Retorna o fator de potência a partir dos valores de potência ativa e aparente já calculados.
- void retornaMEDIACICLOS (float32_t *array_in, float32_t *array_out, uint32_t size, uint32_t n, uint32_t start)

  Tira a média de n ciclos para cálculo de harmônicas.
- void retornaRMSHARMONICOS (float32_t *i_rms_harmonicos, float32_t *array_corrente, uint32_t size, uint32_t h, float32_t g, uint32_t n, uint32_t start)

  Retorna a potência das harmônicas de um vetor com dados de corrente.
- float32_t retornaTHD (float32_t *array_in)

  Retorna o THD a partir de um vetor de harmônicas.

6.17.1 Detailed Description

Implementação das funções que realizam os cálculos dos parâmetros elétricos.

Author

André e Gustavo

6.17.2 Macro Definition Documentation

6.17.2.1 PI_VALUE

```
#define PI_VALUE (float32_t) 3.14159265358
```

6.17.2.2 SQRT2

```
#define SQRT2 (float32_t) 1.414213562373095
```

6.17.3 Function Documentation

6.17.3.1 retornaFP()

```
float32_t retornaFP (
            float32_t potaparente,
            float32_t potativa )
```

Retorna o fator de potência a partir dos valores de potência ativa e aparente já calculados.

Parameters

potaparente	potência aparente.
potativa	potência ativa.

Return values

fp	fator de reativa calculada.

6.17.3.2 retornaMEDIACICLOS()

```
void retornaMEDIACICLOS (
            float32_t * array_in,
            float32_t * array_out,
            uint32_t size,
            uint32_t n,
            uint32_t start )
```

Tira a média de n ciclos para cálculo de harmônicas.

Parameters

array_in	ponteiro com dados de entrada. Tamanho size.
array_out	ponteiro com dados de saída. Tamanho (size-start)/n.
size	número de elementos no vetor de dados.
n	número de ciclos de 60 Hz no vetor de entrada.
start	posição inicial do vetor de dados. Usado para que cálculo ocorra sobre um número inteiro de períodos de 60 Hz.

Return values

None	

6.17.3.3 retornaPOTAPARENTE()

```
float32_t retornaPOTAPARENTE (
            float32_t gv,
            float32_t gi,
            float32_t vrms,
            float32_t irms )
```

Retorna a potência aparente resultante a partir dos valores RMS de tensão e corrente já calculados.

Parameters

gi	ganho para ajuste do valor de corrente (não utilizado).
gv	ganho para ajuste do valor de tensão (não utilizado).
vrms	tensão RMS.
irms	corrente RMS.

Return values

potaparente	potência aparente calculada.

6.17.3.4 retornaPOTATIVA()

```
float32_t retornaPOTATIVA (
            float32_t gv,
            float32_t gi,
            float32_t * array_tensao,
            float32_t * array_corrente,
            uint32_t size,
            uint32_t start )
```

Retorna a potência ativa resultante entre um vetor de tensão e um vetor de corrente.

Parameters

gi	ganho para ajuste do valor de corrente (não utilizado).
gv	ganho para ajuste do valor de tensão (não utilizado).
array_tensao	ponteiro para o vetor de dados de tensão.
array_corrente	ponteiro para o vetor de dados de corrente.
size	número de elementos no vetor de dados.
start	posição inicial do vetor de dados. Usado para que cálculo ocorra sobre um número inteiro de períodos de 60 Hz.

Return values

potativa	potência ativa calculada.

6.17.3.5 retornaPOTREATIVA()

```
float32_t retornaPOTREATIVA (
            float32_t potaparente,
            float32_t potativa )
```

Retorna a potência reativa resultante a partir dos valores de potência ativa e aparente já calculados.

Parameters

potaparente	potência aparente.
potativa	potência ativa.

Return values

potreativa	potência reativa calculada.

6.17.3.6 retornaRMS()

```
float32_t retornaRMS (
            float32_t g,
            float32_t * array,
            uint32_t size,
            uint32_t start )
```

Retorna o valor RMS de um vetor.

Parameters

g	ganho para ajuste do valor (não utilizado).
array	ponteiro para o vetor de dados.
size	número de elementos no vetor de dados.
start	posição inicial do vetor de dados. Usado para que cálculo ocorra sobre um número inteiro de períodos de 60 Hz.

Return values

arrayrms	valor RMS do vetor.

6.17.3.7 retornaRMSHARMONICOS()

```
void retornaRMSHARMONICOS (
            float32_t * i_rms_harmonicos,
            float32_t * array_corrente,
            uint32_t size,
            uint32_t h,
            float32_t g,
            uint32_t n,
            uint32_t start )
```

Retorna a potência das harmônicas de um vetor com dados de corrente.

Parameters

i_rms_harmonicos	ponteiro para vetor de saída com potência das harmônicas.
array_corrente	ponteiro de entrada com valores dos dados de corrente.
g	ganho para ajuste do valor (não utilizado).
size	número de elementos no vetor de dados.
h	número de elementos no vetor de harmônicas.
n	número de ciclos de 60 Hz no vetor de entrada.
start	posição inicial do vetor de dados. Usado para que cálculo ocorra sobre um número inteiro de períodos de 60 Hz.

Return values

None	

6.17.3.8 retornaSIN()

```
void* retornaSIN (
            float32_t * array,
            uint32_t f,
            float32_t fs,
            uint32_t n,
            float32_t phase )
```

DEPRECATED Cria um vetor com sinal senoidal para testes.

Parameters

array	vetor seno
f	frequencia
fs	frequencia de amostragem
n	numero de pontos por periodo
phase	fase do seno

Return values

None	

6.17.3.9 retornaTHD()

```
float32_t retornaTHD (
            float32_t * array_in )
```

Retorna o THD a partir de um vetor de harmônicas.

Parameters

array_in	ponteiro para vetor de harmônicas.

Return values

thd	Distorção Harmônica Total calculadas.

6.18 Src/dma.c File Reference

Header das funções de configuração das DMAs.

```
#include "dma.h"
```

Functions

- void MX_DMA_Init (void)
  Inicialização do canais do DMA 2.

Variables

- DMA_HandleTypeDef hdma_memtomem_dma2_stream1
  Handler para estrutura do stream 1 do DMA 2 para transferência MemToMem.
- DMA_HandleTypeDef hdma_memtomem_dma2_stream3
  Handler para estrutura do stream 3 do DMA 2 para transferência MemToMem.

6.18.1 Detailed Description

Header das funções de configuração das DMAs.

Author

ST

6.18.2 Function Documentation

6.18.2.1 MX_DMA_Init()

```
void MX_DMA_Init (
             void  )
```

Inicialização do canais do DMA 2.

Inicialização dos canais do DMA 2.

Enable DMA controller clock Configure DMA for memory to memory transfers hdma_memtomem_dma2_stream1 hdma_memtomem_dma2_stream3

Return values

None	

6.18.3 Variable Documentation

6.18.3.1 hdma_memtomem_dma2_stream1

```
DMA_HandleTypeDef hdma_memtomem_dma2_stream1
```

Handler para estrutura do stream 1 do DMA 2 para transferência MemToMem.

6.18.3.2 hdma_memtomem_dma2_stream3

```
DMA_HandleTypeDef hdma_memtomem_dma2_stream3
```

Handler para estrutura do stream 3 do DMA 2 para transferência MemToMem.

6.19 Src/equipamentos.c File Reference

Implementação das funções que manipulam as estruturas do projeto.

```
#include "equipamentos.h"
#include "string.h"
```

Functions

- void CadastroDeEquipamento (Equipamento *equip, uint16_t ID, char nome[40], float32_t pot_at, float32_t pot_re, float32_t thd, float32_t pf, float32_t *harmonicos)
  Cria estrutura com informações de um novo equipamento.
- void DeltaParam (Parametros *medida_nova, Parametros *medida_velha, Parametros *delta, char *new)
  Calcula delta entre duas estruturas Parametros .
- float ComparacaoDeEquipamentos (Equipamento *equip1, Parametros *medida)
  DEPRECATED Calcula delta entre duas estruturas Parametros .
- void InitMedicao (Medicao *medicao)
  Inicializa uma estrutura Medicao .
- void InitBaseDeDados (Equipamento *BaseDados)
  Inicializa uma vetor (base de dados) de estruturas Equipamento .
- int IdentificarEquipamento (Equipamento *BaseDados, Parametros *Medicao)
  Função com implementação básica de um algoritmo de identificação de equipamento na rede.

6.19.1 Detailed Description

Implementação das funções que manipulam as estruturas do projeto.

Author

Bruno e Gustavo

6.19.2 Function Documentation

6.19.2.1 CadastroDeEquipamento()

```
void CadastroDeEquipamento (
            Equipamento * equip,
            uint16_t ID,
            char nome[40],
            float32_t pot_at,
            float32_t pot_re,
            float32_t thd,
            float32_t pf,
            float32_t * harmonicos )
```

Cria estrutura com informações de um novo equipamento.

Parameters

equip	ponteiro para estrutura Equipamento .
nome	char∗ com nome do equipamento.
pot_at	potencia ativa.
pot_re	potencia reativa.
thd	taxa de distorção harmonica.
pf	fator de potencia.
harmonicos	vetor de harmonicas.

Return values

None	

6.19.2.2 ComparacaoDeEquipamentos()

```
float ComparacaoDeEquipamentos (
            Equipamento * equip1,
            Parametros * medida )
```

DEPRECATED Calcula delta entre duas estruturas Parametros .

Return values

None	

6.19.2.3 DeltaParam()

```
void DeltaParam (
            Parametros * medida_nova,
            Parametros * medida_velha,
            Parametros * delta,
            char * new )
```

Calcula delta entre duas estruturas Parametros .

Parameters

medida_nova	ponteiro para uma estrutura Parametros que contém os parâmetros da medição mais recente.
medida_velha	ponteiro para uma estrutura Parametros que contém os parâmetros da medição anterior.
delta	ponteiro para uma estrutura Parametros que armazana o delta entre as duas entradas.
new	ponteiro para um char que indicará se um equipamento foi adicionado ou retirado da rede.

Return values

None	

6.19.2.4 IdentificarEquipamento()

```
int IdentificarEquipamento (
            Equipamento * BaseDados,
            Parametros * Medicao )
```

Função com implementação básica de um algoritmo de identificação de equipamento na rede.

Parameters

BaseDados	ponteiro para um vetor de Equipamento sobre o qual é feita a busca da Medicao.
Medicao	ponteiro para Parametros que contém a medição do delta na rede. Representa um equipamento acrescido ou retirado da rede.

Return values

min	Índice da base de dados que contém o equipamento identificado.

6.19.2.5 InitBaseDeDados()

```
void InitBaseDeDados (
            Equipamento * BaseDados )
```

Inicializa uma vetor (base de dados) de estruturas Equipamento .

Parameters

BaseDados	ponteiro para um vetor de Equipamento que será inicializada com zeros.

Return values

None	

6.19.2.6 InitMedicao()

```
void InitMedicao (
            Medicao * medicao )
```

Inicializa uma estrutura Medicao .

Parameters

medicao	ponteiro para uma estrutura Medicao que será inicializada com zeros.

Return values

None	

6.20 Src/gpio.c File Reference

Implementação das funções de configuração do GPIO.

```
#include "gpio.h"
```

Functions

- void MX_GPIO_Init (void)
  Inicialização do GPIO.

6.20.1 Detailed Description

Implementação das funções de configuração do GPIO.

Author

ST

6.20.2 Function Documentation

6.20.2.1 MX_GPIO_Init()

```
void MX_GPIO_Init (
             void  )
```

Inicialização do GPIO.

Return values

None	

6.21 Src/main.c File Reference

Aplicação: Sistema de Monitoramento de Consumo de Energia.

```
#include "main.h"
#include "stm32f4xx_hal.h"
#include "adc.h"
#include "dma.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"
#include "defines.h"
#include "arm_math.h"
#include "dizimacao.h"
#include "adc_util.h"
#include "equipamentos.h"
#include "calculos_eletricos.h"
#include "usart_util.h"
```

Enumerations

- enum FSM {
  START, AQUISICAO, PROCESSAMENTO, RMS_CORRENTE,
  CALCULOS, DELTA, ID, ENVIAR }
  Estados da máquina de estados da aplicação.

Functions

- void SystemClock_Config (void)
  Configuração de clock.
- void Error_Handler (void)
  This function is executed in case of error occurrence.
- void HAL_ADC_ConvCpltCallback (ADC_HandleTypeDef *)
  Callback da interrupção de transferência completa da DMA.
- void HAL_ADC_ConvHalfCpltCallback (ADC_HandleTypeDef *)
  Callback da interrupção de transferência na metade da DMA.
- int main (void)
  Main da aplicação.

Variables

- enum FSM estado = START
- Medicao memoria [MEM_SIZE]

  Vetor de Medicao . Contém histórico de medições.
- uint32_t memoria_index = 0

  Índice do histórico de medições.
- Equipamento BaseDados [EQUIP_ARRAY_MAX]

  Base de dados de Equipamento .
- uint32_t buffer_tensao_DMA [2 ∗BUFFER_SIZE]

  Buffer circular de leitura de tensão. Interrupções ocorrem quando está na metade ou cheio.
- uint32_t buffer_corrente_DMA [2 ∗BUFFER_SIZE]

  Buffer circular de leitura de corrente. Interrupções ocorrem quando está na metade ou cheio.
- uint32_t buffer_tensao_leitura [BUFFER_SIZE]

  Buffer auxiliar de leitura de tensão.
- uint32_t buffer_corrente_leitura [BUFFER_SIZE]

  Buffer auxiliar de leitura de corrente.
- float32_t buffer_corrente_float [BUFFER_SIZE]

  Buffer auxiliar de leitura de tensão em ponto flutuante.
- float32_t buffer_tensao_float [BUFFER_SIZE]

  Buffer auxiliar de leitura de corrente em ponto flutuante.
- float32_t buffer_tensao_diz [BUFFER_DIZ]

  Buffer auxiliar de leitura de tensão em ponto flutuante dizimado.
- float32_t buffer_corrente_diz [BUFFER_DIZ]

  Buffer auxiliar de leitura de corrente em ponto flutuante dizimado.
- uint32_t count = 0

  Variável de testes.
- uint32_t tic

  Variável de testes.
- uint32_t toc
- uint8_t flag_buffercheio = 0

  Flag de controle. Ativada quando ocorre interrupção de buffer cheio.
- uint8_t flag_buffermetade = 0

  Flag de controle. Ativada quando ocorre interrupção de buffer na metade.
- uint8_t flag_aquisicao = 0

  Flag de testes.
- arm_fir_decimate_instance_f32 S

  Estrutura da biblioteca CMSIS para realização da dizimação.

6.21.1 Detailed Description

Aplicação: Sistema de Monitoramento de Consumo de Energia.

Author

André, Bruno, Gustado e Leonador

6.21.2 Enumeration Type Documentation

6.21.2.1 FSM

`enum FSM`

Estados da máquina de estados da aplicação.

Enumerator

START	
AQUISICAO	
PROCESSAMENTO	
RMS_CORRENTE	
CALCULOS	
DELTA	
ID	
ENVIAR	

6.21.3 Function Documentation

6.21.3.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Parameters

None	

Return values

None	

6.21.3.2 HAL_ADC_ConvCpltCallback()

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

Callback da interrupção de transferência completa da DMA.

Parameters

hadc	Handler para ADC.

6.21.3.3 HAL_ADC_ConvHalfCpltCallback()

```
void HAL_ADC_ConvHalfCpltCallback (
             ADC_HandleTypeDef * hadc )
```

Callback da interrupção de transferência na metade da DMA.

Parameters

hadc	Handler para ADC.

6.21.3.4 main()

```
int main (
             void  )
```

Main da aplicação.

Implementa a máquina de estados descrita no relatório 3.

6.21.3.5 SystemClock_Config()

```
void SystemClock_Config (
             void  )
```

Configuração de clock.

System Clock Configuration Configure the main internal regulator output voltage

Initializes the CPU, AHB and APB busses clocks

Initializes the CPU, AHB and APB busses clocks

Configure the Systick interrupt time

Configure the Systick

6.21.4 Variable Documentation

6.21.4.1 BaseDados

```
Equipamento BaseDados[EQUIP_ARRAY_MAX]
```

Base de dados de Equipamento .

6.21.4.2 buffer_corrente_diz

```
float32_t buffer_corrente_diz[BUFFER_DIZ]
```

Buffer auxiliar de leitura de corrente em ponto flutuante dizimado.

6.21.4.3 buffer_corrente_DMA

```
uint32_t buffer_corrente_DMA[2 *BUFFER_SIZE]
```

Buffer circular de leitura de corrente. Interrupções ocorrem quando está na metade ou cheio.

6.21.4.4 buffer_corrente_float

```
float32_t buffer_corrente_float[BUFFER_SIZE]
```

Buffer auxiliar de leitura de tensão em ponto flutuante.

6.21.4.5 buffer_corrente_leitura

```
uint32_t buffer_corrente_leitura[BUFFER_SIZE]
```

Buffer auxiliar de leitura de corrente.

6.21.4.6 buffer_tensao_diz

```
float32_t buffer_tensao_diz[BUFFER_DIZ]
```

Buffer auxiliar de leitura de tensão em ponto flutuante dizimado.

6.21.4.7 buffer_tensao_DMA

```
uint32_t buffer_tensao_DMA[2 *BUFFER_SIZE]
```

Buffer circular de leitura de tensão. Interrupções ocorrem quando está na metade ou cheio.

6.21.4.8 buffer_tensao_float

```
float32_t buffer_tensao_float[BUFFER_SIZE]
```

Buffer auxiliar de leitura de corrente em ponto flutuante.

6.21.4.9 buffer_tensao_leitura

```
uint32_t buffer_tensao_leitura[BUFFER_SIZE]
```

Buffer auxiliar de leitura de tensão.

6.21.4.10 count

```
uint32_t count = 0
```

Variável de testes.

6.21.4.11 estado

```
enum FSM estado = START
```

6.21.4.12 flag_aquisicao

```
uint8_t flag_aquisicao = 0
```

Flag de testes.

6.21.4.13 flag_buffercheio

```
uint8_t flag_buffercheio = 0
```

Flag de controle. Ativada quando ocorre interrupção de buffer cheio.

6.21.4.14 flag_buffermetade

```
uint8_t flag_buffermetade = 0
```

Flag de controle. Ativada quando ocorre interrupção de buffer na metade.

6.21.4.15 memoria

```
Medicao memoria[MEM_SIZE]
```

Vetor de Medicao . Contém histórico de medições.

6.21.4.16 memoria_index

```
uint32_t memoria_index = 0
```

Índice do histórico de medições.

6.21.4.17 S

```
arm_fir_decimate_instance_f32 S
```

Estrutura da biblioteca CMSIS para realização da dizimação.

6.21.4.18 tic

```
uint32_t tic
```

Variável de testes.

6.21.4.19 toc

```
uint32_t toc
```

6.22 Src/stm32f4xx_hal_msp.c File Reference

Inicializa NVIC.

```
#include "stm32f4xx_hal.h"
```

Functions

- void Error_Handler (void)
  This function is executed in case of error occurrence.
- void HAL_MspInit (void)

6.22.1 Detailed Description

Inicializa NVIC.

Author

ST

6.22.2 Function Documentation

6.22.2.1 Error_Handler()

```
void Error_Handler (
            void  )
```

This function is executed in case of error occurrence.

Parameters

None	

Return values

None	

6.22.2.2 HAL_MspInit()

```
void HAL_MspInit (
             void  )
```

Initializes the Global MSP.

6.23 Src/stm32f4xx_it.c File Reference

Interrupt Service Routines.

```
#include "stm32f4xx_hal.h"
#include "stm32f4xx.h"
#include "stm32f4xx_it.h"
```

Functions

- void SVC_Handler (void)

  This function handles System service call via SWI instruction.
- void PendSV_Handler (void)

  This function handles Pendable request for system service.
- void SysTick_Handler (void)

  This function handles System tick timer.
- void ADC_IRQHandler (void)

  This function handles ADC1, ADC2 and ADC3 interrupts.
- void USART2_IRQHandler (void)

  This function handles USART2 global interrupt.
- void DMA2_Stream0_IRQHandler (void)

  This function handles DMA2 stream0 global interrupt.
- void DMA2_Stream1_IRQHandler (void)

  This function handles DMA2 stream1 global interrupt.
- void DMA2_Stream2_IRQHandler (void)

  This function handles DMA2 stream2 global interrupt.
- void DMA2_Stream3_IRQHandler (void)

  This function handles DMA2 stream3 global interrupt.

Variables

- DMA_HandleTypeDef hdma_adc1

  Handler para estrutura do DMA do ADC 1.
- DMA_HandleTypeDef hdma_adc2

  Handler para estrutura do DMA do ADC 2.
- ADC_HandleTypeDef hadc1

  Handler para estrutura do ADC 1.
- ADC_HandleTypeDef hadc2

  Handler para estrutura do ADC 2.
- DMA_HandleTypeDef hdma_memtomem_dma2_stream1

  Handler para estrutura do stream 1 do DMA 2 para transferência MemToMem.
- DMA_HandleTypeDef hdma_memtomem_dma2_stream3

  Handler para estrutura do stream 3 do DMA 2 para transferência MemToMem.
- UART_HandleTypeDef huart2

  Handler para estrutura do UART 2.

6.23.1 Detailed Description

Interrupt Service Routines.

6.23.2 Function Documentation

6.23.2.1 ADC_IRQHandler()

```
void ADC_IRQHandler (
            void  )
```

This function handles ADC1, ADC2 and ADC3 interrupts.

6.23.2.2 DMA2_Stream0_IRQHandler()

```
void DMA2_Stream0_IRQHandler (
            void  )
```

This function handles DMA2 stream0 global interrupt.

6.23.2.3 DMA2_Stream1_IRQHandler()

```
void DMA2_Stream1_IRQHandler (
            void  )
```

This function handles DMA2 stream1 global interrupt.

6.23.2.4 DMA2_Stream2_IRQHandler()

```
void DMA2_Stream2_IRQHandler (
            void  )
```

This function handles DMA2 stream2 global interrupt.

6.23.2.5 DMA2_Stream3_IRQHandler()

```
void DMA2_Stream3_IRQHandler (
            void  )
```

This function handles DMA2 stream3 global interrupt.

6.23.2.6 PendSV_Handler()

```
void PendSV_Handler (
            void  )
```

This function handles Pendable request for system service.

6.23.2.7 SVC_Handler()

```
void SVC_Handler (
            void  )
```

This function handles System service call via SWI instruction.

6.23.2.8 SysTick_Handler()

```
void SysTick_Handler (
            void  )
```

This function handles System tick timer.

6.23.2.9 USART2_IRQHandler()

```
void USART2_IRQHandler (
            void  )
```

This function handles USART2 global interrupt.

6.23.3 Variable Documentation

6.23.3.1 hadc1

```
ADC_HandleTypeDef hadc1
```

Handler para estrutura do ADC 1.

6.23.3.2 hadc2

```
ADC_HandleTypeDef hadc2
```

Handler para estrutura do ADC 2.

6.23.3.3 hdma_adc1

```
DMA_HandleTypeDef hdma_adc1
```

Handler para estrutura do DMA do ADC 1.

6.23.3.4 hdma_adc2

```
DMA_HandleTypeDef hdma_adc2
```

Handler para estrutura do DMA do ADC 2.

6.23.3.5 hdma_memtomem_dma2_stream1

```
DMA_HandleTypeDef hdma_memtomem_dma2_stream1
```

Handler para estrutura do stream 1 do DMA 2 para transferência MemToMem.

6.23.3.6 hdma_memtomem_dma2_stream3

```
DMA_HandleTypeDef hdma_memtomem_dma2_stream3
```

Handler para estrutura do stream 3 do DMA 2 para transferência MemToMem.

6.23.3.7 huart2

```
UART_HandleTypeDef huart2
```

Handler para estrutura do UART 2.

6.24 Src/system_stm32f4xx.c File Reference

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

```
#include "stm32f4xx.h"
```

Macros

- #define HSE_VALUE ((uint32_t)25000000)
- #define HSI_VALUE ((uint32_t)16000000)
- #define VECT_TAB_OFFSET 0x00

Functions

- void SystemInit (void)

  Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.
- void SystemCoreClockUpdate (void)

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

Variables

- uint32_t SystemCoreClock = 16000000
- const uint8_t AHBPrescTable [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t APBPrescTable [8] = {0, 0, 0, 0, 1, 2, 3, 4}

6.24.1 Detailed Description

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

Author

MCD Application Team

Version

V2.6.1

Date

14-February-2017 This file provides two functions and one global variable to be called from user application:

- SystemInit(): This function is called at startup just after reset and before branch to main program. This call is made inside the "startup_stm32f4xx.s" file.
- SystemCoreClock variable: Contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.
- SystemCoreClockUpdate(): Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.

Attention

6.25 Src/tim.c File Reference

Implementação das funções de configuração do temporizador.

```
#include "tim.h"
```

Functions

- void MX_TIM8_Init (void)

  Inicialização dos canais do Timer 8.
- void HAL_TIM_Base_MspInit (TIM_HandleTypeDef *tim_baseHandle)
- void HAL_TIM_Base_MspDeInit (TIM_HandleTypeDef *tim_baseHandle)

Variables

- TIM_HandleTypeDef htim8

  Handler para estrutura do Timer 8.

6.25.1 Detailed Description

Implementação das funções de configuração do temporizador.

Author

ST

6.25.2 Function Documentation

6.25.2.1 HAL_TIM_Base_MspDeInit()

```
void HAL_TIM_Base_MspDeInit (
            TIM_HandleTypeDef * tim_baseHandle )
```

6.25.2.2 HAL_TIM_Base_MspInit()

```
void HAL_TIM_Base_MspInit (
            TIM_HandleTypeDef * tim_baseHandle )
```

6.25.2.3 MX_TIM8_Init()

```
void MX_TIM8_Init (
            void  )
```

Inicialização dos canais do Timer 8.

Return values

None	

6.25.3 Variable Documentation

6.25.3.1 htim8

```
TIM_HandleTypeDef htim8
```

Handler para estrutura do Timer 8.

6.26 Src/usart.c File Reference

Implementação das funções de configuração da uart.

```
#include "usart.h"
#include "gpio.h"
```

Functions

- void MX_USART2_UART_Init (void)

  Inicialização dos canais da UART 2.
- void HAL_UART_MspInit (UART_HandleTypeDef ∗uartHandle)
- void HAL_UART_MspDeInit (UART_HandleTypeDef ∗uartHandle)

Variables

- UART_HandleTypeDef huart2

  Handler para estrutura do UART 2.

6.26.1 Detailed Description

Implementação das funções de configuração da uart.

Author

ST

6.26.2 Function Documentation

6.26.2.1 HAL_UART_MspDeInit()

```
void HAL_UART_MspDeInit (
            UART_HandleTypeDef * uartHandle )
```

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

6.26.2.2 HAL_UART_MspInit()

```
void HAL_UART_MspInit (
            UART_HandleTypeDef * uartHandle )
```

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

6.26.2.3 MX_USART2_UART_Init()

```
void MX_USART2_UART_Init (
            void  )
```

Inicialização dos canais da UART 2.

Return values

None	

6.26.3 Variable Documentation

6.26.3.1 huart2

```
UART_HandleTypeDef huart2
```

Handler para estrutura do UART 2.

6.27 Src/usart_util.c File Reference

Biblioteca com funções úteis para utilizar com a USART2.

```
#include "usart.h"
#include "usart_util.h"
#include "string.h"
#include <stdarg.h>
```

Functions

- void USART2_Transmit_Char (char send)

  Transmite caractere ASCII pela USART 2.
- void USART2_Transmit_String (char ∗send)

  Transmite string pela USART 2.
- void USART2_Transmit_UInt (uint32_t send)

  Transmite inteiro unsigned pela USART 2.
- void USART2_Transmit_Int (int send)

  Transmite inteiro signed pela USART 2.
- void print (const char ∗send,...)

  Função semelhante à printf usando USART 2. Funciona apenas para d e c.
- char USART2_Receive_Command (int timeout)

  Recebe caracter de comando pela USART2.
- void USAR2_Receive_Interrupt_Enable (void)

  Habilita interrupção.

6.27.1 Detailed Description

Biblioteca com funções úteis para utilizar com a USART2.

Author

Gustavo

6.27.2 Function Documentation

6.27.2.1 print()

```
void print (
            const char * send,
             ...  )
```

Função semelhante à printf usando USART 2. Funciona apenas para d e c.

Parameters

send	string formatada a ser transferida pela USART 2

Return values

None	

6.27.2.2 USAR2_Receive_Interrupt_Enable()

```
void USAR2_Receive_Interrupt_Enable (
```

```
            void  )
```

Habilita interrupção.

Return values

None	

6.27.2.3 USART2_Receive_Command()

```
char USART2_Receive_Command (
            int timeout )
```

Recebe caracter de comando pela USART2.

Parameters

timeout	Timeout em ms.

Return values

None	

6.27.2.4 USART2_Transmit_Char()

```
void USART2_Transmit_Char (
            char send )
```

Transmite caractere ASCII pela USART 2.

Parameters

send	caracter a ser transferido pela USART 2

Return values

None	

6.27.2.5 USART2_Transmit_Int()

```
void USART2_Transmit_Int (
            int send )
```

Transmite inteiro signed pela USART 2.

Parameters

send	inteiro a ser transferido pela USART 2

Return values

None	

6.27.2.6 USART2_Transmit_String()

```
void USART2_Transmit_String (
            char * send )
```

Transmite string pela USART 2.

Parameters

send	string a ser transferida pela USART 2

Return values

None	

6.27.2.7 USART2_Transmit_UInt()

```
void USART2_Transmit_UInt (
            uint32_t send )
```

Transmite inteiro unsigned pela USART 2.

Parameters

send	inteiro a ser transferido pela USART 2

Return values

None	

Index