Medidor de temperatura

Generated by Doxygen 1.8.13

Contents

Chapter 1

Medidor de temperatura

Usa-se uma placa SAMR21 conectada a um sensor de temperatura LM35 a fim de que se possa medir a temperatura ambiente. O microcontrolador le os valores de temperatura em intervalos regulares de tempo e calcula as temperaturas maxima, minima, media e atual medidas. As temperaturas medidas sao guardadas dentro da memoria EEPROM do dispositivo. Alem disso, as temperaturas maxima, media, minima e atual sao exibidas em um display OLED com teclado para escolha da informacao desejada. Ademais, os dados de temperatura sao enviados para o computador via interface serial. Ainda, com o auxilio dos botoes do display OLED, pode-se resetar a aplicacao, ou mudar a informacao sendo exibida em tela.

Chapter 2

License

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Module Documentation

6.1 Estados da aplicacao

Macros

- #define NR_STATES 6

  Constante que guarda o numero de estados da aplicacao. Nesse caso, 6 estados.
- #define NR_EVENTS 4

  Constante que guarda o numero de maximo de transicoes da aplicacao. Nesse caso, 4 transicoes.
- #define INIT_SYSTEM 0

  Estado responsavel pela inicializacao dos paratros e metodos essenciais a aplicacao.
- #define LE_SENSOR 1

  Estado que fica lendo a temperatura coletada do sensor periodicamente.
- #define CALC_MEDIA 2

  Estado que calcula a temperatura media, temperatura maxima e temperatura minima, dada a temperatura coletada pelo sensor no estado anterior.
- #define MOSTRA_DISPLAY 3

  Estado que atualiza as informacoes da tela do display mostrando as informacoes.
- #define ESPERA_TAXA 4

  Estado que fica aguardando ate o termino de 2 segundos, quando a aplicacao lera o sensor novamente.
- #define RESET_APPICATION 5

  Estado que reseta todas as informacoes da aplicacao, incluindo a memoria fisica.
- #define MANTEM_ESTADO 0

  Constante que eh usada como evento para a transicao para o proprio estado.
- #define PROXIMO_ESTADO 1

  Constante que eh usada como evento para a transicao para o proximo estado.
- #define ESTADO_ANTERIOR 2

  Constante que eh usada como evento para a transicao para o estado anterior.
- #define HARD_RESET 3

  Constante que eh usada como evento para a transicao para o estado hard_reset.

6.1.1 Detailed Description

6.1.2 Macro Definition Documentation

6.1.2.1 CALC_MEDIA

```
#define CALC_MEDIA 2
```

Estado que calcula a temperatura media, temperatura maxima e temperatura minima, dada a temperatura coletada pelo sensor no estado anterior.

Nesse estado, é usado um vetor de 50 posicoes para o calculo da media dos valores. Após esse estado, a aplicacao vai automaticamente para o estado mostra_display. Para mais informacoes, consulte a funcao de estado calcula_media().

Definition at line 119 of file main.c.

6.1.2.2 ESPERA_TAXA

```
#define ESPERA_TAXA 4
```

Estado que fica aguardando ate o termino de 2 segundos, quando a aplicacao lera o sensor novamente.

Nesse estado, o processador fica em pooling nos botoes do display e no RTC. Se os botoes 1 ou 3 forem pressionados, o processador volta ao estado mostra_display para atualizar a tela, com o estado da FSM modificada pelos botoes. Se o botao 2 for pressionado, o sistema entra no hard reset para reinicializacao total da aplicacao. Ainda, quando o tempo de espera termina, a aplicacao vai para o estado le_sensor novamente. Para mais informacoes, consulte a funcao de estado ocioso().

Definition at line 143 of file main.c.

6.1.2.3 ESTADO_ANTERIOR

```
#define ESTADO_ANTERIOR 2
```

Constante que eh usada como evento para a transicao para o estado anterior.

O proprio estado atribui essa constante a variavel evento.

Definition at line 176 of file main.c.

6.1.2.4 HARD_RESET

```
#define HARD_RESET 3
```

Constante que eh usada como evento para a transicao para o estado hard_reset.

O proprio estado atribui essa constante a variavel evento.

Definition at line 183 of file main.c.

6.1.2.5 INIT_SYSTEM

```
#define INIT_SYSTEM 0
```

Estado responsavel pela inicializacao dos paratros e metodos essenciais a aplicacao.

Nesse estado, sao inicializados os componentes basicos tais como ADC, RTC, OLED, EEPROM e os dados sao recuperados da memoria. Apos esse estado, o sistema vai para o estado le_sensor. Para mais informacoes, consulte a funcao de estado init().

Definition at line 99 of file main.c.

6.1.2.6 LE_SENSOR

```
#define LE_SENSOR 1
```

Estado que fica lendo a temperatura coletada do sensor periodicamente.

Nesse estado, o dado do sensor eh lido com o uso do ADC e uma resolucao de 16 bits. Após esse estado, a aplicacao vai para o estado calcula_media. Para mais informacoes, consulte a funcao de estado le_sensor().

Definition at line 109 of file main.c.

6.1.2.7 MANTEM_ESTADO

```
#define MANTEM_ESTADO 0
```

Constante que eh usada como evento para a transicao para o proprio estado.

O proprio estado atribui essa constante a variavel evento.

Definition at line 161 of file main.c.

6.1.2.8 MOSTRA_DISPLAY

```
#define MOSTRA_DISPLAY 3
```

Estado que atualiza as informacoes da tela do display mostrando as informacoes.

A tela eh atualizada seguindo a maquina de estados interna ao display, a qual eh controlada pelos botoes 1 e 3 do display OLED. Na sequencia desse estado, a aplicacao vai para o estado ocioso(ESPERA_TAXA) Para mais informacoes, ver funcao de estado mostra_display.

Definition at line 129 of file main.c.

6.1.2.9 NR_EVENTS

```
#define NR_EVENTS 4
```

Constante que guarda o numero de maximo de transicoes da aplicacao. Nesse caso, 4 transicoes.

Definition at line 88 of file main.c.

6.1.2.10 NR_STATES

```
#define NR_STATES 6
```

Constante que guarda o numero de estados da aplicacao. Nesse caso, 6 estados.

Definition at line 82 of file main.c.

6.1.2.11 PROXIMO_ESTADO

```
#define PROXIMO_ESTADO 1
```

Constante que eh usada como evento para a transicao para o proximo estado.

O proprio estado atribui essa constante a variavel evento.

Definition at line 169 of file main.c.

6.1.2.12 RESET_APPICATION

```
#define RESET_APPICATION 5
```

Estado que reseta todas as informacoes da aplicacao, incluindo a memoria fisica.

Nesse estado, as informacoes sobre temperatura sao resetadas e o sistema eh completamente reinicializado. Desse modo, o proximo estado eh o init. Para mais informacoes consulte a funcao de estado hard_reset().

Definition at line 153 of file main.c.

6.2 Funcoes dos estados

Functions

- void init ()

  Funcao que representa o estado de inicializacao das variaveis e modulos necessarios para a aplicacao.
- void le_sensor ()

  Funcao que representa o estado de leitura do sensor de temperatura da aplicacao.
- void calcula_media ()

  Funcao que representa o estado de processamento para a obtencao dos valores necessarios para a aplicacao.
- void mostra_display ()

  Funcao que representa o estado de atualizacao do display.
- void ocioso ()

  Funcao que representa o estado de aguardo ate os dois segundos.
- void hard_reset ()

  Funcao que representa o estado de reset total da aplicacao.

6.2.1 Detailed Description

6.2.2 Function Documentation

6.2.2.1 calcula_media()

```
void calcula_media ( )
```

Funcao que representa o estado de processamento para a obtencao dos valores necessarios para a aplicacao.

Pega o valor lido da temperatura atual e coloca no buffer_temp para calcular a media de temperatura, entao calcula a media e grava na eeprom a temperatura_atual, temp_media, temp_min e temp_max, e gera transicao para o estado MOSTRA_DISPLAY. Variavel auxiliar para percorrer o buffer iniciada com a quantidade de dados presentes no buffer

Definition at line 717 of file main.c.

6.2.2.2 hard_reset()

```
void hard_reset ( )
```

Funcao que representa o estado de reset total da aplicacao.

Reseta os valores de temperatura armazenados na memoria eeprom da SAMR21, zera os valores de temperatura_atual, temp_max e temp_media, atribui valor de 255 para temp_min como valores de inicializacao. Apos, gera evento para transicao para o estado init().

Definition at line 870 of file main.c.

6.2.2.3 init()

```
void init ( )
```

Funcao que representa o estado de inicializacao das variaveis e modulos necessarios para a aplicacao.

Estado inicial da aplicacao. O RTC eh configurado para um periodo de 2 segundos, o display OLED eh inicializado, o ADC eh inicializado, as funcoes de escrita no display, sao inicializadas, a maquina de estado de escrita no display eh inicializada, a memoria EEPROM eh inicializada, sao ativadas as interrupcoes para a aplicacao, e os valores das temperaturas atual, media, maxima e minima sao recuperados da memoria EEPROM. Alem disso, o buffer_temp eh zerado e um evento para a transicao para o estado le_sensor eh gerado.

Definition at line 637 of file main.c.

6.2.2.4 le_sensor()

```
void le_sensor ( )
```

Funcao que representa o estado de leitura do sensor de temperatura da aplicacao.

Passa para a placa SAMR21 funcao toggle para piscar o led para sinalizar a medicao. Enquanto isso, o ADC faz a leitura e guarda os valores lidos em adc_result_buffer. Nesse tempo, a aplicacao espera o termino da conversao do ADC. Quando o ADC termina, a funcao faz a media dos valores lidos pelo ADC e mostra no terminal serial o valor medio encontrado. Além disso, o valor medio do ADC eh convertido para uma temperatura em graus Celsius e gera-se um evento para realizar a transicao para o estado CALCULA_MEDIA. Variavel auxiliar para percorrer o buffer com os valores de temperatura lidos pelo sensor

Definition at line 680 of file main.c.

6.2.2.5 mostra_display()

```
void mostra_display ( )
```

Funcao que representa o estado de atualizacao do display.

Atribui a mensagem que cada um dos estados devera mostrar no display, cada estado realiza a conversao do valor "int" lido para "char" possibilitando que este possa ser mostrado no display, e atualiza a tela do display com as informacoes novas. Além disso, manda as informacoes via interface serial e gera um evento para transicionar para o estado ESPERA_TAXA(ocioso).

Definition at line 785 of file main.c.

6.2.2.6 ocioso()

```
void ocioso ( )
```

Funcao que representa o estado de aguardo ate os dois segundos.

Fica em pooling enquanto o timer nao expira e nenhum dos botoes eh pressionado. Se o botao OLED1_BUTTO↩ N1_ID ou OLED1_BUTTON3_ID for pressionado atualiza o estado da maquina de estados para o display e retorna ao estado mostra_display. Caso seja pressionado OLED1_BUTTON2_ID, transiciona para o estado hard_reset() que limpa os dados armazenados ate agora na memoria. Quando o timer expirar, gera evento para a transicao para o estado le_sensor.

Definition at line 843 of file main.c.

6.3 Transicao entre os estados da aplicacao

Data Structures

- struct FSM_STATE_TABLE

  Estrutura usada para transicao entre estados.

Variables

- const FSM_STATE_TABLE StateTable [NR_STATES][NR_EVENTS]
- int evento = 0

  Variavel utilizada para transicoes entre estados da aplicacao Assim, cada funcao de estado seta essa variavel com um dos estados possiveis para a proxima transicao.

6.3.1 Detailed Description

6.3.2 Variable Documentation

6.3.2.1 evento

```
evento = 0
```

Variavel utilizada para transicoes entre estados da aplicacao Assim, cada funcao de estado seta essa variavel com um dos estados possiveis para a proxima transicao.

Definition at line 271 of file main.c.

6.3.2.2 StateTable

```
const FSM_STATE_TABLE StateTable[NR_STATES][NR_EVENTS]
```

Initial value:

```
=
{
      init, INIT_SYSTEM,                          init,
      LE_SENSOR,                    init, LE_SENSOR,
      init, RESET_APPICATION,
      le_sensor, LE_SENSOR,                       le_sensor,
      CALC_MEDIA,             le_sensor, CALC_MEDIA,
      le_sensor, RESET_APPICATION,
      calcula_media, CALC_MEDIA,
      calcula_media, MOSTRA_DISPLAY,
      calcula_media, MOSTRA_DISPLAY,       calcula_media,
      RESET_APPICATION,
      mostra_display, MOSTRA_DISPLAY,
      mostra_display, ESPERA_TAXA,         mostra_display,
      ESPERA_TAXA,       mostra_display, RESET_APPICATION,
      ocioso, ESPERA_TAXA,                          ocioso,
      LE_SENSOR,                  ocioso, MOSTRA_DISPLAY,
      ocioso, RESET_APPICATION,
      hard_reset, INIT_SYSTEM,
      hard_reset, INIT_SYSTEM,             hard_reset,
      INIT_SYSTEM,            hard_reset, RESET_APPICATION,
}
```

Matriz que contem a relacao entre os estados e possiveis transicoes para a aplicacao

Definition at line 256 of file main.c.

6.4 Modulos da aplicacao

Modules

- USART
- OLED
- RTC
- ADC
- EEPROM

6.4.1 Detailed Description

6.5 USART

Data Structures

- struct usart_instance

  Estrutura usada para usar a interface serial da aplicacao.
- struct usart_conf

  Estrutura para a configuracao do modulo usart para a comunicacao serial.

Variables

- struct usart_module usart_instance
- struct usart_config usart_conf

6.5.1 Detailed Description

6.5.2 Variable Documentation

6.5.2.1 usart_conf

```
struct usart_config usart_conf
```

Instancia da estrutura usart_config para a configuracao da USART

Definition at line 290 of file main.c.

6.5.2.2 usart_instance

```
struct usart_module usart_instance
```

Instancia da estrutura usart_module

Definition at line 286 of file main.c.

6.6 OLED

Enumerations

- enum state { TEMP_ATUAL = 0, TEMP_MEDIA, TEMP_MAX, TEMP_MIN }

  Enumerador para transicao entre estados internos do display OLED.

Variables

- int x

  Variavel que marca a posicao x de escrita no display.
- int y

  Variavel que marca a posicao y de escrita no display.
- char c [50]

  Variavel que recebe o valor de temperaturas convertidas para string (itoa)
- char mensagem [20]

  Variavel que recebe as mensagens temperatura atual, temperatura media, etc para a escrita no display.
- enum state estado

6.6.1 Detailed Description

6.6.2 Enumeration Type Documentation

6.6.2.1 state

```
enum state
```

Enumerador para transicao entre estados internos do display OLED.

Essa variavel controla qual sera a informacao mostrada no display OLED por uma maquina de estados propria e eh mudada de acordo com os botoes do display OLED (botoes 1 e 3).

Enumerator

TEMP_ATUAL	
TEMP_MEDIA	
TEMP_MAX	
TEMP_MIN	

Definition at line 330 of file main.c.

6.6.3 Variable Documentation

6.6.3.1 c

`c`

Variavel que recebe o valor de temperaturas convertidas para string (itoa)

Definition at line 316 of file main.c.

6.6.3.2 estado

`enum state estado`

Guarda o estado para mostrar no display

6.6.3.3 mensagem

`mensagem`

Variavel que recebe as mensagens temperatura atual, temperatura media, etc para a escrita no display.

Definition at line 321 of file main.c.

6.6.3.4 x

`x`

Variavel que marca a posicao x de escrita no display.

Definition at line 306 of file main.c.

6.6.3.5 y

`y`

Variavel que marca a posicao y de escrita no display.

Definition at line 311 of file main.c.

6.7 RTC

Data Structures

- struct rtc_instance

  Estrutura responsavel pela representacao do timer criado.

Functions

- void configure_rtc_count (void)

  Funcao que configura o RTC (timer) para um tempo de 2 segundos.

Variables

- struct rtc_module rtc_instance

6.7.1 Detailed Description

6.7.2 Function Documentation

6.7.2.1 configure_rtc_count()

```
void configure_rtc_count (
             void  )
```

Funcao que configura o RTC (timer) para um tempo de 2 segundos.

Configuracao RTC com prescaler de 32 (divide por 32 o clock) e modo de contagem de 16 bits. Ainda, o periodo do timer eh setado como 2 segundos. Alem disso, essa funcao inicializa o timer e torna ele ativo.

Definition at line 602 of file main.c.

6.7.3 Variable Documentation

6.7.3.1 rtc_instance

```
struct rtc_module rtc_instance
```

Guarda a instancia do RTC criada

Definition at line 347 of file main.c.

6.8 ADC

Data Structures

- struct adc_instance

  Estrutura responsavel pela representacao do ADC configurado.

Macros

- #define ADC_SAMPLES 128

  Constante define quantos valores o ADC ira coletar a cada operacao. Depois sera feita uma media desses valores.

Functions

- void configure_adc (void)

  Funcao que configura o ADC para o uso da aplicacao.
- void configure_adc_callbacks (void)

  Funcao que configura as interrupcoes para o ADC.
- void adc_complete_callback (struct adc_module ∗const)

Variables

- struct adc_module adc_instance
- uint16_t adc_result_buffer [ADC_SAMPLES]

  Variavel que recebe as conversoes do ADC.
- volatile bool adc_read_done = false

  Variavel que indica que o ADC terminou uma conversao.

6.8.1 Detailed Description

6.8.2 Macro Definition Documentation

6.8.2.1 ADC_SAMPLES

```
#define ADC_SAMPLES 128
```

Constante define quantos valores o ADC ira coletar a cada operacao. Depois sera feita uma media desses valores.

Definition at line 360 of file main.c.

6.8.3 Function Documentation

6.8.3.1 adc_complete_callback()

```
void adc_complete_callback (
            struct adc_module * const module )
```

Definition at line 486 of file main.c.

6.8.3.2 configure_adc()

```
void configure_adc (
            void  )
```

Funcao que configura o ADC para o uso da aplicacao.

Funcao que inicializa os parametros do ADC como default. Depois, define a referencia do ADC como 1.7 V, define o PRESCALER como dividido por 8, define uma resolucao de 16 bits e seta o pino PB02 (ADC[10]) como entrada positiva. A entrada negativa eh aterrada e o modo de operacao eh definido como single-ended. Depois, ativa o ADC.

Definition at line 499 of file main.c.

6.8.3.3 configure_adc_callbacks()

```
void configure_adc_callbacks (
            void  )
```

Funcao que configura as interrupcoes para o ADC.

Habilita as interrupcoes geradas pelo ADC dado o modulo inicializado anteriormente.

Definition at line 530 of file main.c.

6.8.4 Variable Documentation

6.8.4.1 adc_instance

```
struct adc_module adc_instance
```

Guarda a instancia do ADC configurada

Definition at line 370 of file main.c.

6.8.4.2 adc_read_done

```
adc_read_done = false
```

Variavel que indica que o ADC terminou uma conversao.

Definition at line 391 of file main.c.

6.8.4.3 adc_result_buffer

```
adc_result_buffer
```

Variavel que recebe as conversoes do ADC.

Definition at line 386 of file main.c.

6.9 EEPROM

Functions

- void configure_eeprom (void)

  Funcao que configura a memoria EEPROM para uso.

Variables

- uint8_t page_data [EEPROM_PAGE_SIZE]

  Buffer auxiliar para escrita e leitura da memoria EEPROM.

6.9.1 Detailed Description

6.9.2 Function Documentation

6.9.2.1 configure_eeprom()

```
void configure_eeprom (
            void  )
```

Funcao que configura a memoria EEPROM para uso.

Inicializa o emulador da EEPROM e verifica se ha erros. Dois tipos de erros podem ocorrer: 1) Sem area de memoria: Nesse caso, a aplicacao entre em loop infinito e nao faz mais nada. 2) Memoria corrompida: Nesse caso, a aplicacao reseta a memoria e a reinicia. Caso nenhum dos erros ocorra, a memoria comeca a funcionar corretamente.

Definition at line 543 of file main.c.

6.9.3 Variable Documentation

6.9.3.1 page_data

```
page_data
```

Buffer auxiliar para escrita e leitura da memoria EEPROM.

Definition at line 408 of file main.c.

6.10 Informacoes temperatura

Macros

- #define TAM_BUFFER 50

  Constante que marca o tamanho do buffer que sera usado para calculo da media.

Variables

- uint8_t temperatura_atual

  Variavel que guarda a temperatura atual medida.
- uint8_t temp_media

  Variavel que guarda a temperatura media das temperaturas atuais medidas.
- uint8_t temp_max

  Variavel que guarda a temperatura maxima medida.
- uint8_t temp_min

  Variavel que guarda a temperatura minima medida.
- uint16_t conversao_temperatura

  Variavel que guarda a media dos valores lidos pelo ADC (16 bits)
- volatile uint8_t buffer_temp [TAM_BUFFER]

  Buffer que guarda as ultimas temperaturas atuais para calculo da media.
- volatile uint8_t i_buffer = 0

  Guarda o inicio do buffer_temp.
- volatile uint8_t f_buffer = 0

  Guarda o final do buffer_temp.
- volatile uint8_t cont_buffer = 0

  Guarda o numero de posicoes preenchidas do buffer_temp.

6.10.1 Detailed Description

6.10.2 Macro Definition Documentation

6.10.2.1 TAM_BUFFER

```
#define TAM_BUFFER 50
```

Constante que marca o tamanho do buffer que sera usado para calculo da media.

Definition at line 448 of file main.c.

6.10.3 Variable Documentation

6.10.3.1 buffer_temp

```
buffer_temp
```

Buffer que guarda as ultimas temperaturas atuais para calculo da media.

Definition at line 453 of file main.c.

6.10.3.2 cont_buffer

```
cont_buffer = 0
```

Guarda o numero de posicoes preenchidas do buffer_temp.

Definition at line 468 of file main.c.

6.10.3.3 conversao_temperatura

```
conversao_temperatura
```

Variavel que guarda a media dos valores lidos pelo ADC (16 bits)

Definition at line 443 of file main.c.

6.10.3.4 f_buffer

```
f_buffer = 0
```

Guarda o final do buffer_temp.

Definition at line 463 of file main.c.

6.10.3.5 i_buffer

```
i_buffer = 0
```

Guarda o inicio do buffer_temp.

Definition at line 458 of file main.c.

6.10.3.6 temp_max

`temp_max`

Variavel que guarda a temperatura maxima medida.

Definition at line 433 of file main.c.

6.10.3.7 temp_media

`temp_media`

Variavel que guarda a temperatura media das temperaturas atuais medidas.

Definition at line 428 of file main.c.

6.10.3.8 temp_min

`temp_min`

Variavel que guarda a temperatura minima medida.

Definition at line 438 of file main.c.

6.10.3.9 temperatura_atual

`temperatura_atual`

Variavel que guarda a temperatura atual medida.

Definition at line 423 of file main.c.

Chapter 7

Data Structure Documentation

7.1 adc_instance Struct Reference

Estrutura responsavel pela representacao do ADC configurado.

7.1.1 Detailed Description

Estrutura responsavel pela representacao do ADC configurado.

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

- src/main.c

7.2 FSM_STATE_TABLE Struct Reference

Estrutura usada para transicao entre estados.

Data Fields

- void(∗ ptrFunc)(void)

  Ponteiro para a funcao do estado atual.
- uint8_t NextState

  Variavel para transicao de estados que guarda o proximo estado.

7.2.1 Detailed Description

Estrutura usada para transicao entre estados.

A figura abaixo mostra as transicoes entre os estados da aplicacao.

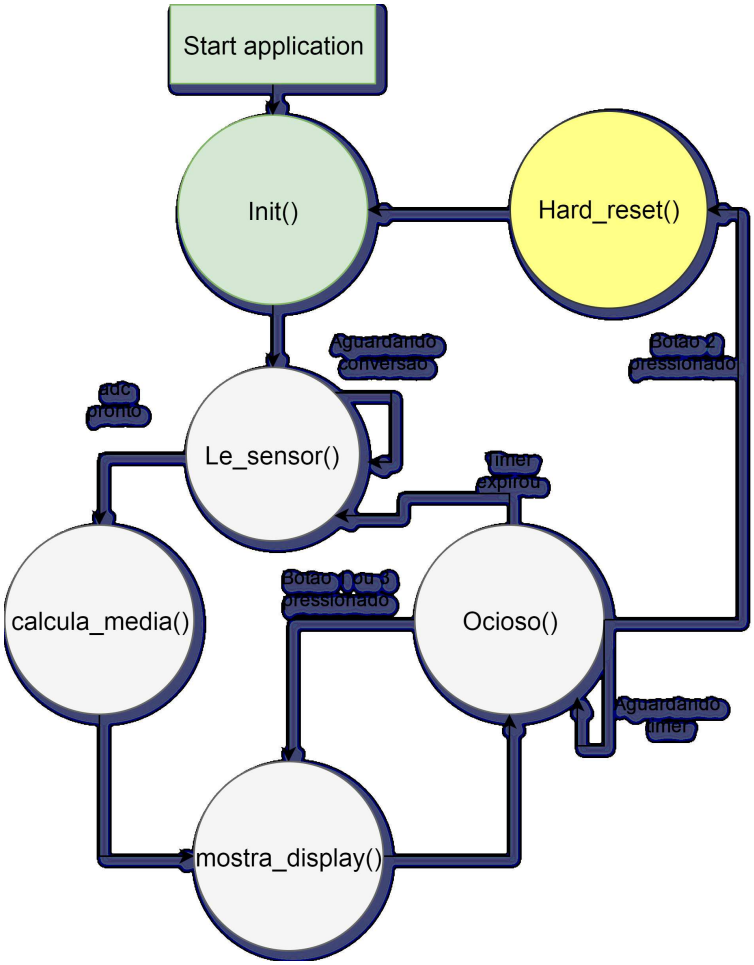

Figure 7.1 Diagrama de transicao de estados

Essa estrutura guarda um parametro para a funcao do determinado estado e um sinal para possiveis transicoes desse estado para outros estados.

Definition at line 240 of file main.c.

7.2.2 Field Documentation

7.2.2.1 NextState

`NextState`

Variavel para transicao de estados que guarda o proximo estado.

Definition at line 250 of file main.c.

7.2.2.2 ptrFunc

`ptrFunc`

Ponteiro para a funcao do estado atual.

Definition at line 245 of file main.c.

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

- src/main.c

7.3 rtc_instance Struct Reference

Estrutura responsavel pela representacao do timer criado.

7.3.1 Detailed Description

Estrutura responsavel pela representacao do timer criado.

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

- src/main.c

7.4 usart_conf Struct Reference

Estrutura para a configuracao do modulo usart para a comunicacao serial.

7.4.1 Detailed Description

Estrutura para a configuracao do modulo usart para a comunicacao serial.

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

- src/main.c

7.5 usart_instance Struct Reference

Estrutura usada para usar a interface serial da aplicacao.

7.5.1 Detailed Description

Estrutura usada para usar a interface serial da aplicacao.

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

- src/main.c

Chapter 8

File Documentation

8.1 src/asf.h File Reference

Autogenerated API include file for the Atmel Software Framework (ASF)

```
#include <adc.h>
#include <adc_callback.h>
#include <bod.h>
#include <compiler.h>
#include <status_codes.h>
#include <delay.h>
#include <eeprom.h>
#include <gfx_mono.h>
#include <sysfont.h>
#include <board.h>
#include <interrupt.h>
#include <nvm.h>
#include <port.h>
#include <parts.h>
#include <rtc_count.h>
#include <rtc_tamper.h>
#include <sercom.h>
#include <sercom_interrupt.h>
#include <spi.h>
#include <spi_interrupt.h>
#include <usart.h>
#include <usart_interrupt.h>
#include <ssd1306.h>
#include <clock.h>
#include <gclk.h>
#include <system.h>
#include <pinmux.h>
#include <system_interrupt.h>
#include <power.h>
#include <reset.h>
#include <stdio_serial.h>
#include <serial.h>
#include <cdc.h>
#include <oled1.h>
```

8.1.1 Detailed Description

Autogenerated API include file for the Atmel Software Framework (ASF)

8.2 src/main.c File Reference

Arquivo principal do projeto com todas as funções criadas.

```
#include <asf.h>
#include <stdio.h>
#include <conf_demo.h>
#include "oled1.h"
#include <math.h>
```

Data Structures

- struct FSM_STATE_TABLE

  Estrutura usada para transicao entre estados.

Macros

- #define NR_STATES 6

  Constante que guarda o numero de estados da aplicacao. Nesse caso, 6 estados.
- #define NR_EVENTS 4

  Constante que guarda o numero de maximo de transicoes da aplicacao. Nesse caso, 4 transicoes.
- #define INIT_SYSTEM 0

  Estado responsavel pela inicializacao dos paratros e metodos essenciais a aplicacao.
- #define LE_SENSOR 1

  Estado que fica lendo a temperatura coletada do sensor periodicamente.
- #define CALC_MEDIA 2

  Estado que calcula a temperatura media, temperatura maxima e temperatura minima, dada a temperatura coletada pelo sensor no estado anterior.
- #define MOSTRA_DISPLAY 3

  Estado que atualiza as informacoes da tela do display mostrando as informacoes.
- #define ESPERA_TAXA 4

  Estado que fica aguardando ate o termino de 2 segundos, quando a aplicacao lera o sensor novamente.
- #define RESET_APPICATION 5

  Estado que reseta todas as informacoes da aplicacao, incluindo a memoria fisica.
- #define MANTEM_ESTADO 0

  Constante que eh usada como evento para a transicao para o proprio estado.
- #define PROXIMO_ESTADO 1

  Constante que eh usada como evento para a transicao para o proximo estado.
- #define ESTADO_ANTERIOR 2

  Constante que eh usada como evento para a transicao para o estado anterior.
- #define HARD_RESET 3

  Constante que eh usada como evento para a transicao para o estado hard_reset.
- #define ADC_SAMPLES 128

  Constante define quantos valores o ADC ira coletar a cada operacao. Depois sera feita uma media desses valores.
- #define TAM_BUFFER 50

  Constante que marca o tamanho do buffer que sera usado para calculo da media.

Enumerations

- enum state { TEMP_ATUAL = 0, TEMP_MEDIA, TEMP_MAX, TEMP_MIN }

  Enumerador para transicao entre estados internos do display OLED.

Functions

- void init ()

  Funcao que representa o estado de inicializacao das variaveis e modulos necessarios para a aplicacao.
- void le_sensor ()

  Funcao que representa o estado de leitura do sensor de temperatura da aplicacao.
- void calcula_media ()

  Funcao que representa o estado de processamento para a obtencao dos valores necessarios para a aplicacao.
- void mostra_display ()

  Funcao que representa o estado de atualizacao do display.
- void ocioso ()

  Funcao que representa o estado de aguardo ate os dois segundos.
- void hard_reset ()

  Funcao que representa o estado de reset total da aplicacao.
- void configure_rtc_count (void)

  Funcao que configura o RTC (timer) para um tempo de 2 segundos.
- void configure_adc (void)

  Funcao que configura o ADC para o uso da aplicacao.
- void configure_adc_callbacks (void)

  Funcao que configura as interrupcoes para o ADC.
- void adc_complete_callback (struct adc_module *const)
- void configure_eeprom (void)

  Funcao que configura a memoria EEPROM para uso.
- int main (void)

  Funcao principal do programa para inicializacoes e troca de estados.
- void SYSCTRL_Handler (void)
- void debounce ()

  Debounce para os botoes Rotina que gasta um pequeno tempo para o ajuste do debounce dos botoes do display OLED.

Variables

- const FSM_STATE_TABLE StateTable [NR_STATES][NR_EVENTS]
- int evento = 0

  Variavel utilizada para transicoes entre estados da aplicacao Assim, cada funcao de estado seta essa variavel com um dos estados possiveis para a proxima transicao.
- struct usart_module usart_instance
- struct usart_config usart_conf
- int x

  Variavel que marca a posicao x de escrita no display.
- int y

  Variavel que marca a posicao y de escrita no display.
- char c [50]

  Variavel que recebe o valor de temperaturas convertidas para string (itoa)
- char mensagem [20]

Variavel que recebe as mensagens temperatura atual, temperatura media, etc para a escrita no display.

- enum state estado
- struct rtc_module rtc_instance
- struct adc_module adc_instance
- uint16_t adc_result_buffer [ADC_SAMPLES]

  Variavel que recebe as conversoes do ADC.
- volatile bool adc_read_done = false

  Variavel que indica que o ADC terminou uma conversao.
- uint8_t page_data [EEPROM_PAGE_SIZE]

  Buffer auxiliar para escrita e leitura da memoria EEPROM.
- uint8_t temperatura_atual

  Variavel que guarda a temperatura atual medida.
- uint8_t temp_media

  Variavel que guarda a temperatura media das temperaturas atuais medidas.
- uint8_t temp_max

  Variavel que guarda a temperatura maxima medida.
- uint8_t temp_min

  Variavel que guarda a temperatura minima medida.
- uint16_t conversao_temperatura

  Variavel que guarda a media dos valores lidos pelo ADC (16 bits)
- volatile uint8_t buffer_temp [TAM_BUFFER]

  Buffer que guarda as ultimas temperaturas atuais para calculo da media.
- volatile uint8_t i_buffer = 0

  Guarda o inicio do buffer_temp.
- volatile uint8_t f_buffer = 0

  Guarda o final do buffer_temp.
- volatile uint8_t cont_buffer = 0

  Guarda o numero de posicoes preenchidas do buffer_temp.

8.2.1 Detailed Description

Arquivo principal do projeto com todas as funções criadas.

Author

Grégory e Vinícius

Date

28/06/2017

8.2.2 Function Documentation

8.2.2.1 debounce()

```
debounce ( )
```

Debounce para os botoes Rotina que gasta um pequeno tempo para o ajuste do debounce dos botoes do display OLED.

Definition at line 623 of file main.c.

8.2.2.2 main()

```
int main (
            void  )
```

Funcao principal do programa para inicializacoes e troca de estados.

Inicializa o sistema, usa a comunicacao serial da placa, seta os pinos da placa que serao usados no uso serial, nesse caso serao usados o EDBG_CDC_SERCOM_PINMUX_PAD0, EDBG_CDC_SERCOM_PINMUX_PA↩D1, EDBG_CDC_SERCOM_PINMUX_PAD2, EDBG_CDC_SERCOM_PINMUX_PAD3. Alem disso, seta o estado inicial como INIT_SYSTEM, e continua a transicao e troca de estados da aplicacao. currentState eh uma variavel (local) para guardar o estado atual da aplicacao.

Definition at line 904 of file main.c.

8.2.2.3 SYSCTRL_Handler()

```
SYSCTRL_Handler (
            void  )
```

Interrupcao para a atualizacao da pagina na memoria nao-volatil acionada pelo BOD.

Definition at line 567 of file main.c.

Index