Referência do Arquivo driver_init.c

```
#include "driver_init.h"
#include <peripheral_clk_config.h>
#include <utils.h>
#include <hal_init.h>
#include <hpl_gclk_base.h>
#include <hpl_pm_base.h>
#include <hpl_adc_base.h>
```

Definições e Macros

#define	**IO_SENSOR_ADC_CH_AMOUNT** 1
#define	**IO_SENSOR_ADC_BUFFER_SIZE** 16
#define	**IO_SENSOR_ADC_CH_MAX** 0

Funções

void **IO_SENSOR_ADC_INIT** (void)
Inicializa o controlador ADC e configura o pino PA06 para uso com o ADC.

void **TARGETIO_PORT_INIT** (void)
Inicializa o controlador USART (configura os pinos PA04 e PA05 para uso com USART).

void **TARGETIO_CLOCK_INIT** (void)
Inicializa o clock para o m�dulo SERCOM0. Geralmente é usado para comunica��o serial no microcontrolador.

void **TARGETIO_INIT** (void)
Inicializa o m�dulo de comunica��o USART (Universal Synchronous and Asynchronous Receiver-Transmitter) usando o SERCOM0.

void **I2C_INSTANCE_PORT_INIT** (void)
Inicializa o controlador I2C (configura os pinos PA16 e PA17 para uso com I2C).

void **I2C_INSTANCE_CLOCK_INIT** (void)
Inicializa o clock para o m�dulo SERCOM1.

void **I2C_INSTANCE_INIT** (void)
Inicializa o controlador de comunica��o I2C usando o m�dulo SERCOM1.

void **delay_driver_init** (void)
Inicializa o driver de atraso usando SysTick.

void **system_init** (void)
Inicializa o sistema, configurando o microcontrolador e chamando as fun��es de inicializa��o dos controladores.

Variáveis

struct adc_async_descriptor **IO_SENSOR_ADC**
Controlador ADC ass�ncrono para o sensor de luz.

struct adc_async_channel_descriptor	**IO_SENSOR_ADC_ch** [**IO_SENSOR_ADC_CH_AMOUNT**]
struct usart_sync_descriptor	**TARGETIO** Descritor de comunica��o USART s�ncrona.
struct i2c_m_sync_desc	**I2C_INSTANCE** Descritor de comunica��o I2C mestre s�ncrona.

Definições e macros

◆ IO_SENSOR_ADC_BUFFER_SIZE

```
#define IO_SENSOR_ADC_BUFFER_SIZE   16
```

◆ IO_SENSOR_ADC_CH_AMOUNT

```
#define IO_SENSOR_ADC_CH_AMOUNT   1
```

◆ IO_SENSOR_ADC_CH_MAX

```
#define IO_SENSOR_ADC_CH_MAX   0
```

Funções

◆ delay_driver_init()

```
void delay_driver_init ( void   )
```

Inicializa o driver de atraso usando SysTick.

```
 99                                {
100     delay_init(SysTick);
101 }
```

◆ I2C_INSTANCE_CLOCK_INIT()

void I2C_INSTANCE_CLOCK_INIT (void)

Inicializa o clock para o m□dulo SERCOM1.

```
81                                      {
82      _pm_enable_bus_clock(PM_BUS_APBC, SERCOM1);
83      _gclk_enable_channel(SERCOM1_GCLK_ID_CORE, CONF_GCLK_SERCOM1_CORE_SRC);
84      _gclk_enable_channel(SERCOM1_GCLK_ID_SLOW, CONF_GCLK_SERCOM1_SLOW_SRC);
85  }
```

◆ I2C_INSTANCE_INIT()

void I2C_INSTANCE_INIT (void)

Inicializa o controlador de comunica□□o I2C usando o m□dulo SERCOM1.

Inicializa o controlador de comunica□□o I2C.

```
90                                  {
91      I2C_INSTANCE_CLOCK_INIT();
92      i2c_m_sync_init(&I2C_INSTANCE, SERCOM1);
93      I2C_INSTANCE_PORT_INIT();
94  }
```

◆ I2C_INSTANCE_PORT_INIT()

void I2C_INSTANCE_PORT_INIT (void)

Inicializa o controlador I2C (configura os pinos PA16 e PA17 para uso com I2C).

Inicializa os pinos necess□rios para a comunica□□o I2C.

```
68                                      {
69      gpio_set_pin_pull_mode(PA16, GPIO_PULL_OFF);
70      gpio_set_pin_function(PA16, PINMUX_PA16C_SERCOM1_PAD0);
71      gpio_set_pin_pull_mode(PA17, GPIO_PULL_OFF);
72      gpio_set_pin_function(PA17, PINMUX_PA17C_SERCOM1_PAD1);
73  }
```

◆ IO_SENSOR_ADC_INIT()

void IO_SENSOR_ADC_INIT (void)

Inicializa o controlador ADC e configura o pino PA06 para uso com o ADC.

Inicializa o controlador ADC para o sensor de luz.

```
27                                  {
28      _pm_enable_bus_clock(PM_BUS_APBC, ADC);
29      _gclk_enable_channel(ADC_GCLK_ID, CONF_GCLK_ADC_SRC);
30      adc_async_init(&IO_SENSOR_ADC, ADC, IO_SENSOR_ADC_map, IO_SENSOR_ADC_CH_MAX,
31      IO_SENSOR_ADC_CH_AMOUNT, &IO_SENSOR_ADC_ch[0], (void *)NULL);
32      adc_async_register_channel_buffer(&IO_SENSOR_ADC, 0, IO_SENSOR_ADC_buffer,
33      IO_SENSOR_ADC_BUFFER_SIZE);
34      gpio_set_pin_direction(PA06, GPIO_DIRECTION_OFF);
35      gpio_set_pin_function(PA06, PINMUX_PA06B_ADC_AIN6);
36  }
```

◆ system_init()

void system_init (void)

Inicializa o sistema, configurando o microcontrolador e chamando as fun��es de inicializa��o dos controladores.

Perform system initialization, initialize pins and clocks for peripherals.

```
106                             {
107     init_mcu(); // Inicializa o microcontrolador
108     //Configura o pino do LED e o coloca em n�vel baixo.
109     gpio_set_pin_level(LED, false);
110     gpio_set_pin_direction(LED, GPIO_DIRECTION_OUT);
111     gpio_set_pin_function(LED, GPIO_PIN_FUNCTION_OFF);
112     //Chama as fun��es de inicializa��o dos controladores
113     IO_SENSOR_ADC_INIT();
114     TARGETIO_INIT();
115     I2C_INSTANCE_INIT();
116     delay_driver_init();
117 }
```

◆ TARGETIO_CLOCK_INIT()

void TARGETIO_CLOCK_INIT (void)

Inicializa o clock para o m�dulo SERCOM0. Geralmente é usado para comunica��o serial no microcontrolador.

Inicializa o clock para o m�dulo SERCOM0 usado para USART.

```
50                                  {
51      _pm_enable_bus_clock(PM_BUS_APBC, SERCOM0);
52      _gclk_enable_channel(SERCOM0_GCLK_ID_CORE, CONF_GCLK_SERCOM0_CORE_SRC);
53  }
```

◆ TARGETIO_INIT()

void TARGETIO_INIT (void)

Inicializa o m�dulo de comunica��o USART (Universal Synchronous and Asynchronous Receiver-Transmitter) usando o SERCOM0.

```
59                          {
60      TARGETIO_CLOCK_INIT();
61      usart_sync_init(&TARGETIO, SERCOM0, (void *)NULL);
62      TARGETIO_PORT_INIT();
63  }
```

◆ TARGETIO_PORT_INIT()

void TARGETIO_PORT_INIT (void)

Inicializa o controlador USART (configura os pinos PA04 e PA05 para uso com USART).

Inicializa os pinos necess�rios para a comunica��o USART.

```
41                              {
42      gpio_set_pin_function(PA04, PINMUX_PA04D_SERCOM0_PAD0);
43      gpio_set_pin_function(PA05, PINMUX_PA05D_SERCOM0_PAD1);
44  }
```

Variáveis

◆ I2C_INSTANCE

struct i2c_m_sync_desc I2C_INSTANCE

Descritor de comunica��o I2C mestre s�ncrona.

◆ IO_SENSOR_ADC

struct adc_async_descriptor IO_SENSOR_ADC

Controlador ADC ass�ncrono para o sensor de luz.

◆ IO_SENSOR_ADC_ch

struct adc_async_channel_descriptor IO_SENSOR_ADC_ch[**IO_SENSOR_ADC_CH_AMOUNT**]

◆ TARGETIO

struct usart_sync_descriptor TARGETIO

Descritor de comunica��o USART s�ncrona.

Referência do Arquivo driver_init.h

```
#include "atmel_start_pins.h"
#include <hal_atomic.h>
#include <hal_delay.h>
#include <hal_gpio.h>
#include <hal_init.h>
#include <hal_io.h>
#include <hal_sleep.h>
#include <hal_adc_async.h>
#include <hal_usart_sync.h>
#include <hal_i2c_m_sync.h>
```

Ir para o código-fonte desse arquivo.

Funções

void	**IO_SENSOR_ADC_INIT** (void) Inicializa o controlador ADC para o sensor de luz.
void	**TARGETIO_PORT_INIT** (void) Inicializa os pinos necess&#65533;rios para a comunica&#65533;&#65533;o USART.
void	**TARGETIO_CLOCK_INIT** (void) Inicializa o clock para o m&#65533;dulo SERCOM0 usado para USART.
void	**TARGETIO_init** (void) Inicializa o controlador de comunica&#65533;&#65533;o USART.
void	**I2C_INSTANCE_CLOCK_INSTANCE** (void) Inicializa o clock para o m&#65533;dulo SERCOM1 usado para I2C.
void	**I2C_INSTANCE_INIT** (void) Inicializa o controlador de comunica&#65533;&#65533;o I2C.
void	**I2C_INSTANCE_PORT_INIT** (void) Inicializa os pinos necess&#65533;rios para a comunica&#65533;&#65533;o I2C.
void	**delay_driver_init** (void) Inicializa o driver de atraso usando SysTick.
void	**system_init** (void) Perform system initialization, initialize pins and clocks for peripherals.

Variáveis

struct adc_async_descriptor	**IO_SENSOR_ADC** Controlador ADC ass&#65533;ncrono para o sensor de luz.
struct usart_sync_descriptor	**TARGETIO** Descritor de comunica&#65533;&#65533;o USART s&#65533;ncrona.
struct i2c_m_sync_desc	**I2C_INSTANCE**

Descritor de comunica��o I2C mestre s�ncrona.

Funções

◆ delay_driver_init()

void delay_driver_init (void)

Inicializa o driver de atraso usando SysTick.

```
 99                                {
100     delay_init(SysTick);
101 }
```

◆ I2C_INSTANCE_CLOCK_INSTANCE()

void I2C_INSTANCE_CLOCK_INSTANCE (void)

Inicializa o clock para o m�dulo SERCOM1 usado para I2C.

◆ I2C_INSTANCE_INIT()

void I2C_INSTANCE_INIT (void)

Inicializa o controlador de comunica��o I2C.

Inicializa o controlador de comunica��o I2C.

```
 90                                {
 91     I2C_INSTANCE_CLOCK_INIT();
 92     i2c_m_sync_init(&I2C_INSTANCE, SERCOM1);
 93     I2C_INSTANCE_PORT_INIT();
 94 }
```

◆ I2C_INSTANCE_PORT_INIT()

void I2C_INSTANCE_PORT_INIT (void)

Inicializa os pinos necess�rios para a comunica��o I2C.

Inicializa os pinos necess�rios para a comunica��o I2C.

```
68                                      {
69      gpio_set_pin_pull_mode(PA16, GPIO_PULL_OFF);
70      gpio_set_pin_function(PA16, PINMUX_PA16C_SERCOM1_PAD0);
71      gpio_set_pin_pull_mode(PA17, GPIO_PULL_OFF);
72      gpio_set_pin_function(PA17, PINMUX_PA17C_SERCOM1_PAD1);
73  }
```

◆ IO_SENSOR_ADC_INIT()

void IO_SENSOR_ADC_INIT (void)

Inicializa o controlador ADC para o sensor de luz.

Inicializa o controlador ADC para o sensor de luz.

```
27                                  {
28      _pm_enable_bus_clock(PM_BUS_APBC, ADC);
29      _gclk_enable_channel(ADC_GCLK_ID, CONF_GCLK_ADC_SRC);
30      adc_async_init(&IO_SENSOR_ADC, ADC, IO_SENSOR_ADC_map, IO_SENSOR_ADC_CH_MAX,
31      IO_SENSOR_ADC_CH_AMOUNT, &IO_SENSOR_ADC_ch[0], (void *)NULL);
32      adc_async_register_channel_buffer(&IO_SENSOR_ADC, 0, IO_SENSOR_ADC_buffer,
33      IO_SENSOR_ADC_BUFFER_SIZE);
34      gpio_set_pin_direction(PA06, GPIO_DIRECTION_OFF);
35      gpio_set_pin_function(PA06, PINMUX_PA06B_ADC_AIN6);
36  }
```

◆ system_init()

void system_init (void)

Perform system initialization, initialize pins and clocks for peripherals.

Perform system initialization, initialize pins and clocks for peripherals.

```
106                             {
107     init_mcu(); // Inicializa o microcontrolador
108     //Configura o pino do LED e o coloca em n�vel baixo.
109     gpio_set_pin_level(LED, false);
110     gpio_set_pin_direction(LED, GPIO_DIRECTION_OUT);
111     gpio_set_pin_function(LED, GPIO_PIN_FUNCTION_OFF);
112     //Chama as fun��es de inicializa��o dos controladores
113     IO_SENSOR_ADC_INIT();
114     TARGETIO_INIT();
115     I2C_INSTANCE_INIT();
116     delay_driver_init();
117 }
```

TARGETIO_CLOCK_INIT()

void TARGETIO_CLOCK_INIT (void)

Inicializa o clock para o m�dulo SERCOM0 usado para USART.

Inicializa o clock para o m�dulo SERCOM0 usado para USART.

```
50                                  {
51      _pm_enable_bus_clock(PM_BUS_APBC, SERCOM0);
52      _gclk_enable_channel(SERCOM0_GCLK_ID_CORE, CONF_GCLK_SERCOM0_CORE_SRC);
53  }
```

TARGETIO_init()

void TARGETIO_init (void)

Inicializa o controlador de comunica��o USART.

TARGETIO_PORT_INIT()

void TARGETIO_PORT_INIT (void)

Inicializa os pinos necess�rios para a comunica��o USART.

Inicializa os pinos necess�rios para a comunica��o USART.

```
41                                  {
42      gpio_set_pin_function(PA04, PINMUX_PA04D_SERCOM0_PAD0);
43      gpio_set_pin_function(PA05, PINMUX_PA05D_SERCOM0_PAD1);
44  }
```

Variáveis

I2C_INSTANCE

struct i2c_m_sync_desc I2C_INSTANCE extern

Descritor de comunica��o I2C mestre s�ncrona.

IO_SENSOR_ADC

struct adc_async_descriptor IO_SENSOR_ADC extern

Controlador ADC ass◆ncrono para o sensor de luz.

◆ TARGETIO

struct usart_sync_descriptor TARGETIO extern

Descritor de comunica◆◆o USART s◆ncrona.

Referência do Arquivo IO1_drivers.c

Fun��es relacionadas ao sensor de luz e suas opera��es. Mais...

```
#include <IO1_drivers.h>
```

Funções

void	**IO_SENSOR_INIT** (void) Inicializa os par�metros do sensor de luz, da USART e do sensor de temperatura.
void	**sendByteToUART** (uint8_t byte_to_send) Envia um byte para a UART de debug.
float	**readVoltageSensor** (void) L� o valor digital do sensor de luz ap�s passar pelo ADC e calcula a tens�o medida pelo sensor.
float	**readCurrentSensor** (float voltage) Calcula a corrente com base na diferen�a entre a tens�o de refer�ncia e a tens�o medida, considerando a rela��o entre a corrente do fototransistor e a ilumin�ncia.
float	**readLightSensor** (float current) Calcula a ilumin�ncia com base nos valores medidos de corrente.
uint16_t	**readTemperatureSensor** (void) L� a temperatura do sensor de temperatura.
void	**IO1X_LED_ON** (void) Liga o LED da placa de expans�o IO1X.
void	**floatToString** (float num, char *str, int precision) Converte um n�mero float em string com a precis�o informada.

Variáveis

volatile bool	**conversion_done** = false Flag que indica se a convers�o anal�gico-digital do sensor de luz foi conclu�da.

Descrição detalhada

Fun��es relacionadas ao sensor de luz e suas opera��es.

Funções

◆ floatToString()

```
void floatToString ( float   num,
                     char *  str,
                     int     precision
                   )
```

Converte um n�mero float em string com a precis�o informada.

Parâmetros

num N�mero a ser convertido.

str String resultante da convers�o.

precision Precis�o da convers�o.

```
117                                                     {
118     int i = 0;
119     int integralPart = (int)num;
120 
121     /* Converte a parte inteira para string */
122     do {
123         str[i++] = integralPart % 10 + '0';
124         integralPart /= 10;
125     } while (integralPart > 0);
126 
127     /* Inverte a string da parte inteira */
128     int j;
129     int len = i;
130     for (j = 0; j < len / 2; j++) {
131         char temp = str[j];
132         str[j] = str[len - j - 1];
133         str[len - j - 1] = temp;
134     }
135 
136     /* Adiciona ponto decimal */
137     str[i++] = '.';
138 
139     /* Converte a parte fracion�ria para string */
140     float fractionalPart = num - (int)num;
141     int k;
142     for (k = 0; k < precision; k++) {
143         fractionalPart *= 10;
144         int digit = (int)fractionalPart;
145         str[i++] = digit + '0';
146         fractionalPart -= digit;
147     }
148 
149     /* Adiciona caractere de t�rmino */
150     str[i] = '\0';
151 }
```

◆ IO1X_LED_ON()

void IO1X_LED_ON (void)

Liga o LED da placa de expans&#65533;o IO1X.

```
106                         {
107     gpio_set_pin_level(LED, false);
108 }
```

IO_SENSOR_INIT()

void IO_SENSOR_INIT (void)

Inicializa os par&#65533;metros do sensor de luz, da USART e do sensor de temperatura.

Inicializa os par&#65533;metros do sensor de luz, USART e sensor de temperatura.

```
25                          {
26     /* Inicializa&#65533;&#65533;o par&#65533;metros sensor de luz */
27     adc_async_register_callback(&IO_SENSOR_ADC, 0, ADC_ASYNC_CONVERT_CB,
    lightSensor_ADC_conversion_callback);
28     adc_async_enable_channel(&IO_SENSOR_ADC, 0);
29     adc_async_start_conversion(&IO_SENSOR_ADC);
30 
31     /* Inicializa&#65533;&#65533;o par&#65533;metros USART */
32     usart_sync_get_io_descriptor(&TARGETIO, &TARGETIO_DEBUG);
33 
34     /* Inicializa&#65533;&#65533;o par&#65533;metros do sensor de temperatura */
35     i2c_m_sync_enable(&I2C_INSTANCE);
36     AT30TSE75X = at30tse75x_construct(&AT30TSE75X_descr.parent, &I2C_INSTANCE,
    CONF_AT30TSE75X_RESOLUTION);
37 }
```

readCurrentSensor()

float readCurrentSensor (float voltage)

Calcula a corrente com base na diferen&#65533;a entre a tens&#65533;o de refer&#65533;ncia e a tens&#65533;o medida, considerando a rela&#65533;&#65533;o entre a corrente do fototransistor e a ilumin&#65533;ncia.

Parâmetros

voltage Tens&#65533;o medida pelo sensor de luz.

Retorna

Corrente calculada.

```
73                                      {
74     float IO_SENSOR_CURRENT;
75     /* Calcula a corrente com base na diferen&#65533;a entre a tens&#65533;o de refer&#65533;ncia e a
    tens&#65533;o medida,
76      considerando a rela&#65533;&#65533;o entre a corrente do fototransistor e a ilumin&#65533;ncia*/
77     IO_SENSOR_CURRENT = (VCC_TARGET - voltage)/100000;
78     return IO_SENSOR_CURRENT ;
79 }
```

◆ readLightSensor()

float readLightSensor (float current)

Calcula a ilumin&#65533;ncia com base nos valores medidos de corrente.

Parâmetros

current Corrente medida.

Retorna

Ilumin&#65533;ncia calculada.

```
87                                      {
88     float IO_ILUMINANCE;    //Representa a ilumin&#65533;ncia calculada com base nos valores
       medidos
89
90     IO_ILUMINANCE = (current * 2 *10)/0.000001;
91     return IO_ILUMINANCE;
92 }
```

◆ readTemperatureSensor()

uint16_t readTemperatureSensor (void)

L&#65533; a temperatura do sensor de temperatura.

Retorna

Temperatura lida.

```
99                                      {
100     return (uint16_t)temperature_sensor_read(AT30TSE75X);
101 }
```

◆ readVoltageSensor()

float readVoltageSensor (void)

L□ o valor digital do sensor de luz ap□s passar pelo ADC e calcula a tens□o medida pelo sensor.

Retorna

Tens□o medida pelo sensor de luz.

```
53                                {
54      uint8_t IO_SENSOR_VALUE;  //Armazena o valor lido do sensor de luz
55      float IO_SENSOR_VOLTAGE; //Armazena a tens□o medida pelo sensor
56      /* Faz a convers□o AD do sensor de luz*/
57      adc_async_start_conversion(&IO_SENSOR_ADC);
58      while(!conversion_done){}
59      adc_async_read_channel(&IO_SENSOR_ADC, 0, &IO_SENSOR_VALUE, 1);
60
61      /* Faz a defini□□o dos valores de tens□o lidos do sensor a partir dos dados
    quantizados do ADC*/
62      IO_SENSOR_VOLTAGE = IO_SENSOR_VALUE * VCC_TARGET / 255;
63      return IO_SENSOR_VOLTAGE;
64  }
```

◆ sendByteToUART()

void sendByteToUART (uint8_t byte_to_send)

Envia um byte para a UART de debug.

Parâmetros

byte_to_send Byte a ser enviado.

```
44                                          {
45      io_write(TARGETIO_DEBUG, &byte_to_send, 1);
46  }
```

Variáveis

◆ conversion_done

volatile bool conversion_done = false

Flag que indica se a convers□o anal□gico-digital do sensor de luz foi conclu□da.

Referência do Arquivo IO1_drivers.h

```
#include "driver_init.h"
#include <stdio.h>
#include <at30tse75x.h>
#include <temperature_sensor.h>
#include <at30tse75x_config.h>
```

Ir para o código-fonte desse arquivo.

Definições e Macros

#define	**VCC_TARGET** 3.3
#define	**CONF_AT30TSE75X_RESOLUTION** 2

Funções

void	**IO_SENSOR_INIT** (void) Inicializa os par&#65533;metros do sensor de luz, USART e sensor de temperatura.
void	**sendByteToUART** (uint8_t byte_to_send) Envia um byte para a UART de debug.
float	**readVoltageSensor** (void) L&#65533; o valor digital do sensor de luz ap&#65533;s passar pelo ADC e calcula a tens&#65533;o medida pelo sensor.
float	**readCurrentSensor** (float voltage) Calcula a corrente com base na diferen&#65533;a entre a tens&#65533;o de refer&#65533;ncia e a tens&#65533;o medida, considerando a rela&#65533;&#65533;o entre a corrente do fototransistor e a ilumin&#65533;ncia.
float	**readLightSensor** (float current) Calcula a ilumin&#65533;ncia com base nos valores medidos de corrente.
void	**IO1X_LED_ON** (void) Liga o LED da placa de expans&#65533;o IO1X.
void	**IO1X_LED_OFF** (void) Desliga o LED da placa de expans&#65533;o IO1X.
void	**floatToString** (float num, char *str, int precision) Converte um n&#65533;mero float em string com a precis&#65533;o informada.

Variáveis

struct **temperature_sensor** *	**AT30TSE75X**
struct io_descriptor *	**TARGETIO_DEBUG**

Definições e macros

◆ CONF_AT30TSE75X_RESOLUTION

```
#define CONF_AT30TSE75X_RESOLUTION  2
```

Resolu��o configurada para o sensor de temperatura

◆ VCC_TARGET

```
#define VCC_TARGET  3.3
```

Tens�o VCC da placa de R21 usada como referencia

Funções

◆ floatToString()

```
void floatToString ( float  num,
                     char * str,
                     int    precision
                   )
```

Converte um n�mero float em string com a precis�o informada.

Parâmetros

num	N�mero a ser convertido.
str	String resultante da convers�o.
precision	Precis�o da convers�o.

Converte um número float em string com a precisão informada É usada para poder imprimir um valor float na tela usando o sprintf O compilador usado não aceita float no sprintf

```
117                                                     {
118     int i = 0;
119     int integralPart = (int)num;
120 
121     /* Converte a parte inteira para string */
122     do {
123         str[i++] = integralPart % 10 + '0';
124         integralPart /= 10;
125     } while (integralPart > 0);
126 
127     /* Inverte a string da parte inteira */
128     int j;
129     int len = i;
130     for (j = 0; j < len / 2; j++) {
131         char temp = str[j];
132         str[j] = str[len - j - 1];
133         str[len - j - 1] = temp;
134     }
135 
136     /* Adiciona ponto decimal */
137     str[i++] = '.';
138 
139     /* Converte a parte fracion�ria para string */
140     float fractionalPart = num - (int)num;
141     int k;
142     for (k = 0; k < precision; k++) {
143         fractionalPart *= 10;
144         int digit = (int)fractionalPart;
145         str[i++] = digit + '0';
146         fractionalPart -= digit;
147     }
148 
149     /* Adiciona caractere de t�rmino */
150     str[i] = '\0';
151 }
```

◆ IO1X_LED_OFF()

void IO1X_LED_OFF (void)

Desliga o LED da placa de expans◆o IO1X.

IO1X_LED_ON()

void IO1X_LED_ON (void)

Liga o LED da placa de expans◆o IO1X.

```
106                         {
107     gpio_set_pin_level(LED, false);
108 }
```

IO_SENSOR_INIT()

void IO_SENSOR_INIT (void)

Inicializa os par◆metros do sensor de luz, USART e sensor de temperatura.

Inicializa os par◆metros do sensor de luz, USART e sensor de temperatura.

```
25                          {
26     /* Inicializa◆◆o par◆metros sensor de luz */
27     adc_async_register_callback(&IO_SENSOR_ADC, 0, ADC_ASYNC_CONVERT_CB,
    lightSensor_ADC_conversion_callback);
28     adc_async_enable_channel(&IO_SENSOR_ADC, 0);
29     adc_async_start_conversion(&IO_SENSOR_ADC);
30 
31     /* Inicializa◆◆o par◆metros USART */
32     usart_sync_get_io_descriptor(&TARGETIO, &TARGETIO_DEBUG);
33 
34     /* Inicializa◆◆o par◆metros do sensor de temperatura */
35     i2c_m_sync_enable(&I2C_INSTANCE);
36     AT30TSE75X = at30tse75x_construct(&AT30TSE75X_descr.parent, &I2C_INSTANCE,
    CONF_AT30TSE75X_RESOLUTION);
37 }
```

readCurrentSensor()

float readCurrentSensor (float voltage)

Calcula a corrente com base na diferen�a entre a tens�o de refer�ncia e a tens�o medida, considerando a rela��o entre a corrente do fototransistor e a ilumin�ncia.

Parâmetros

voltage Tens�o medida pelo sensor de luz.

Retorna

Corrente calculada.

```
73                                      {
74      float IO_SENSOR_CURRENT;
75      /* Calcula a corrente com base na diferen�a entre a tens�o de refer�ncia e a
    tens�o medida,
76       considerando a rela��o entre a corrente do fototransistor e a ilumin�ncia*/
77      IO_SENSOR_CURRENT = (VCC_TARGET - voltage)/100000;
78      return IO_SENSOR_CURRENT ;
79  }
```

◆ readLightSensor()

float readLightSensor (float current)

Calcula a ilumin�ncia com base nos valores medidos de corrente.

Parâmetros

current Corrente medida.

Retorna

Ilumin�ncia calculada.

```
87                                      {
88      float IO_ILUMINANCE;    //Representa a ilumin�ncia calculada com base nos valores
    medidos
89
90      IO_ILUMINANCE = (current * 2 *10)/0.000001;
91      return IO_ILUMINANCE;
92  }
```

◆ readVoltageSensor()

float readVoltageSensor (void)

L□ o valor digital do sensor de luz ap□s passar pelo ADC e calcula a tens□o medida pelo sensor.

Retorna

Tens□o medida pelo sensor de luz.

```
53                                {
54      uint8_t IO_SENSOR_VALUE;  //Armazena o valor lido do sensor de luz
55      float IO_SENSOR_VOLTAGE; //Armazena a tens□o medida pelo sensor
56      /* Faz a convers□o AD do sensor de luz*/
57      adc_async_start_conversion(&IO_SENSOR_ADC);
58      while(!conversion_done){}
59      adc_async_read_channel(&IO_SENSOR_ADC, 0, &IO_SENSOR_VALUE, 1);
60
61      /* Faz a defini□□o dos valores de tens□o lidos do sensor a partir dos dados
    quantizados do ADC*/
62      IO_SENSOR_VOLTAGE = IO_SENSOR_VALUE * VCC_TARGET / 255;
63      return IO_SENSOR_VOLTAGE;
64  }
```

◆ sendByteToUART()

void sendByteToUART (uint8_t byte_to_send)

Envia um byte para a UART de debug.

Parâmetros

byte_to_send Byte a ser enviado.

```
44                                          {
45      io_write(TARGETIO_DEBUG, &byte_to_send, 1);
46  }
```

Variáveis

◆ AT30TSE75X

struct **temperature_sensor*** AT30TSE75X

Descritor para o sensor de temperatura AT30TSE75X

◆ TARGETIO_DEBUG

struct io_descriptor* TARGETIO_DEBUG

Descritor para a interface de comunica��o USART de debug

Referência do Arquivo main.c

C�digo de teste dos sensores da placa IO1X Plained. Mais...

```
#include <atmel_start.h>
#include <stdio.h>
#include <IO1_drivers.h>
```

Funções

int **main** (void)
Fun��o principal.

Descrição detalhada

C�digo de teste dos sensores da placa IO1X Plained.

Desenvolvimento atual por Kalidsa Buzzatti de Oliveira.

Funções

◆ main()

int main (void)

Fun��o principal.

< Inclui os arquivos de fun��o dos sensores da placa IO1X Plained.

Inicializa o MCU, drivers e middleware, e realiza a leitura dos sensores em um loop continuo.

```
18 {
19     /* Initializes MCU, drivers and middleware */
20     atmel_start_init();
21     IO_SENSOR_INIT();
22 
23     char message[15]; // Mensagem a ser enviada pela serial para o terminal
24 
25     while (1) {
26 
27             //Liga LED da placa de expans�o
28             IO1X_LED_ON();
29 
30              // Leitura e envio da ilumin�ncia calculada
31             float voltageSensor = readVoltageSensor();
32             char voltage_str[20];
33             floatToString(voltageSensor, voltage_str, 4);
34             sprintf(message, "Tens�o do sensor: %s V\r\n",voltage_str);
35             printf(message);
36 
37             float currentSensor = readCurrentSensor(voltageSensor);
38             char current_str[20];
39             floatToString(currentSensor, current_str, 8);
40             sprintf(message, "Corrente no sensor: %s �mpere\r\n", current_str);
41             printf(message);
42 
43             float light_sensor = readLightSensor(currentSensor); // Ilumin�ncia
   medida pelo sensor de luz
44             delay_ms(100);
45             char iluminance_str[20];
46             floatToString(light_sensor, iluminance_str, 4);
47 
48             sprintf(message, "Iluminancia: %s lux\r\n\n", iluminance_str);
49             printf(message);
50 
51             // Leitura e envio da temperatura
52             uint16_t temperature = readTemperatureSensor(); // Temperatura medida
   pelo sensor
53             sprintf(message, "Temperatura: %d C\r\n", temperature);
54             printf(message);
55     }
56 }
```