HAL Team - Source code documentation - IO1X

1.16

Generated by Doxygen 1.9.7

Chapter 1

License

Subject to your compliance with these terms, you may use Microchip software and any derivatives exclusively with Microchip products. It is your responsibility to comply with third party license terms applicable to your use of third party software (including open source software) that may accompany Microchip software.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP ¨AS IS¨. NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED WARRANTIES OF NON-↩ INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

\asf_license_stop

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 temperature_sensor Struct Reference

Abstract temperature sensor.

`#include <temperature_sensor.h>`

Data Fields

- void * io
- const struct temperature_sensor_interface * interface

4.1.1 Detailed Description

Abstract temperature sensor.

4.1.2 Field Documentation

4.1.2.1 interface

```
const struct temperature_sensor_interface* interface
```

The interface of abstract temperature sensor

4.1.2.2 io

```
void* io
```

The pointer to interface used to communicate with temperature sensor

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

- ot-rtos_elc1048-main/HAL_Sensores/HAL_Io1_Plained/Test_Io1_Plained/temperature_sensor.h

4.2 temperature_sensor_interface Struct Reference

Interface of abstract temperature sensor.

`#include <temperature_sensor.h>`

Data Fields

- float(∗ **read**)(const struct temperature_sensor ∗const me)

4.2.1 Detailed Description

Interface of abstract temperature sensor.

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

- ot-rtos_elc1048-main/HAL_Sensores/HAL_Io1_Plained/Test_Io1_Plained/temperature_sensor.h

Chapter 5

File Documentation

5.1 ot-rtos_elc1048-main/HAL_Sensores/HAL_Io1_Plained/Test_Io1_↩ Plained/at30tse75x.c File Reference

AT30TSE75X driver.

```
#include <at30tse75x.h>
#include <hal_i2c_m_sync.h>
#include <temperature_sensor.h>
#include <at30tse75x_config.h>
```

Macros

- #define AT30TSE_TEMPERATURE_REG 0
- #define **AT30TSE_CONFIGURATION_REG** 1
- #define AT30TSE_NON_VOLATILE_REG_TYPE 0
- #define AT30TSE_SENSOR_ADDRESS 0x4F
- #define AT30TSE_CONFIGURATION_REG_RESOLUTION_BF_OFFSET 13

Functions

- struct temperature_sensor * **at30tse75x_construct** (struct temperature_sensor *const me, void *const io, const uint8_t resolution)

  Construct at30tse75x temperature sensor.
- float **at30tse75x_read** (const struct temperature_sensor *const me)

  Read temperature from the given sensor.

5.1.1 Detailed Description

AT30TSE75X driver.

Copyright (c) 2016-2018 Microchip Technology Inc. and its subsidiaries.

\asf_license_start

5.1.2 Macro Definition Documentation

5.1.2.1 AT30TSE_CONFIGURATION_REG_RESOLUTION_BF_OFFSET

```
#define AT30TSE_CONFIGURATION_REG_RESOLUTION_BF_OFFSET 13
```

Offset of resolution bit-field

5.1.2.2 AT30TSE_NON_VOLATILE_REG_TYPE

```
#define AT30TSE_NON_VOLATILE_REG_TYPE 0
```

Register types of at30tse75x

5.1.2.3 AT30TSE_SENSOR_ADDRESS

```
#define AT30TSE_SENSOR_ADDRESS 0x4F
```

Address of temperature sensor

5.1.2.4 AT30TSE_TEMPERATURE_REG

```
#define AT30TSE_TEMPERATURE_REG 0
```

Register addresses of at30tse75x

5.2 ot-rtos_elc1048-main/HAL_Sensores/HAL_Io1_Plained/Test_Io1_↩ Plained/IO1X_Plained_drivers.c File Reference

Temperature Sensor implementation.

```
#include <IO1X_Plained_drivers.h>
```

Functions

- void sensors_init (void)
- void UART_write_byte (uint8_t byte_to_send)

  Writes a byte on the debugger UART.
- float **GET_light_sensor** (void)

  Reads and process the digital value of the ADC Outputs the iluminance, based on the sensor current vs iluminance curve.
- uint16_t **GET_temperature_sensor** (void)
- void SET_IO1X_LED_ON (void)

  Turns the IO1X board LED on.
- void SET_IO1X_LED_OFF (void)

  Turns the IO1X board LED off.
- void **floatToString** (float num, char ∗str, int precision)

  Converts a float number into string with a specified precision The compiler does not accept a float for printf, so it needs to be converted.

Variables

- volatile bool **conversion_done** = false

5.2.1 Detailed Description

Temperature Sensor implementation.

5.2.2 Function Documentation

5.2.2.1 sensors_init()

```
void sensors_init (
            void  )
```

ADC parameters initialization

USART parameters initialization

Temperature sensor parameters initialization

5.2.2.2 SET_IO1X_LED_OFF()

```
void SET_IO1X_LED_OFF (
            void  )
```

Turns the IO1X board LED off.

Turn the IO1X expansion board LED off.

5.2.2.3 SET_IO1X_LED_ON()

```
void SET_IO1X_LED_ON (
            void  )
```

Turns the IO1X board LED on.

Turn the IO1X expansion board LED on.

5.2.2.4 UART_write_byte()

```
void UART_write_byte (
            uint8_t byte_to_send )
```

Writes a byte on the debugger UART.

Writes a byte in the debugging UART.

5.3 ot-rtos_elc1048-main/HAL_Sensores/HAL_Io1_Plained/Test_Io1_↩ Plained/IO1X_Plained_drivers.h File Reference

IO1X Extension board drivers.

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

Macros

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

Functions

- void sensors_init (void)
- void UART_write_byte (uint8_t byte_to_send)

  Writes a byte on the debugger UART.
- float **GET_light_sensor** (void)

  Reads and process the digital value of the ADC Outputs the iluminance, based on the sensor current vs iluminance curve.
- void SET_IO1X_LED_ON (void)

  Turns the IO1X board LED on.
- void SET_IO1X_LED_OFF (void)

  Turns the IO1X board LED off.
- void **floatToString** (float num, char ∗str, int precision)

  Converts a float number into string with a specified precision The compiler does not accept a float for printf, so it needs to be converted.

Variables

- struct temperature_sensor ∗ **AT30TSE75X**
- struct io_descriptor ∗ **USART_debug_io**

5.3.1 Detailed Description

IO1X Extension board drivers.

5.3.2 Macro Definition Documentation

5.3.2.1 VCC_TARGET

```
#define VCC_TARGET 3.3
```

Parameters

VCC_TARGET	R21 board VCC voltage used as reference

5.3.3 Function Documentation

5.3.3.1 sensors_init()

```
void sensors_init (
             void  )
```

ADC parameters initialization

USART parameters initialization

Temperature sensor parameters initialization

5.3.3.2 SET_IO1X_LED_OFF()

```
void SET_IO1X_LED_OFF (
             void  )
```

Turns the IO1X board LED off.

Turn the IO1X expansion board LED off.

5.3.3.3 SET_IO1X_LED_ON()

```
void SET_IO1X_LED_ON (
             void  )
```

Turns the IO1X board LED on.

Turn the IO1X expansion board LED on.

5.3.3.4 UART_write_byte()

```
void UART_write_byte (
             uint8_t byte_to_send )
```

Writes a byte on the debugger UART.

Writes a byte in the debugging UART.

5.4 IO1X_Plained_drivers.h

Go to the documentation of this file.

```
00001
00005 #ifndef _IO1X_Plained_drivers
00006 #define _IO1X_Plained_drivers
00007
00008 #include "driver_init.h"
00009 #include <stdio.h>
00010 #include <at30tse75x.h>
00011 #include <temperature_sensor.h>
00012 #include <at30tse75x_config.h>
00013
00017 #define VCC_TARGET 3.3
00018 #define CONF_AT30TSE75X_RESOLUTION 2
00019
00020 struct temperature_sensor *AT30TSE75X;
00021 struct io_descriptor* USART_debug_io;
00022
00023 static struct at30tse75x AT30TSE75X_descr;
00024
00025 void sensors_init(void);
00026 void UART_write_byte(uint8_t byte_to_send);
00027 float GET_light_sensor(void);
00028 void SET_IO1X_LED_ON(void);
00029 void SET_IO1X_LED_OFF(void);
00030 void floatToString(float num, char* str, int precision);
00031
00032
00033 #endif
```

5.5 ot-rtos_elc1048-main/HAL_Sensores/HAL_Io1_Plained/Test_Io1_↩ Plained/main.c File Reference

Sensors test code∗.

```
#include <atmel_start.h>
#include <stdio.h>
#include <IO1X_Plained_drivers.h>
```

Macros

- #define Test_temperature_sensor
- #define Test_sensor_light
- #define Test_LED

5.5.1 Detailed Description

Sensors test code∗.

Developed by Guilherme Ribeiro Silveira

Calculation process link: https://docs.google.com/document/d/14jGzlbBa0Ey1Iu↩ W73ailFBDOAaCUMx8I5gUdWG36onQ/edit?usp=sharing

5.5.2 Macro Definition Documentation

5.5.2.1 Test_LED

```
#define Test_LED
```

LED test

5.5.2.2 Test_sensor_light

```
#define Test_sensor_light
```

Light sensor test, displaying on serial terminal

5.5.2.3 Test_temperature_sensor

```
#define Test_temperature_sensor
```

Temperature sensor test, displaying on serial terminal

5.6 ot-rtos_elc1048-main/HAL_Sensores/HAL_Io1_Plained/Test_Io1_↩ Plained/TempAcele_e_Iluini_main.h File Reference

Temperature Sensor implementation.

```
#include <atmel_start.h>
#include <stdio.h>
#include <bno055.h>
```

Macros

- #define VCC_TARGET 3.3

Functions

- void **Light_sensor_init** (void)
- void **USART_dbg_init** (void)
- void UART_write_byte (uint8_t byte_to_send)

  Writes a byte on the debugger UART.
- float **GET_light_sensor** (void)

  Reads and process the digital value of the ADC Outputs the iluminance, based on the sensor current vs iluminance curve.
- void SET_IO1X_LED_ON (void)

  Turns the IO1X board LED on.
- void SET_IO1X_LED_OFF (void)

  Turns the IO1X board LED off.
- void **floatToString** (float num, char ∗str, int precision)

  Converts a float number into string with a specified precision The compiler does not accept a float for printf, so it needs to be converted.
- int **main** (void)

Variables

- volatile bool **conversion_done** = false
- struct io_descriptor * **USART_debug_io**
- struct bno055_dev **bno055**
- float **acceleration_x**
- float **acceleration_y**
- float **acceleration_z**
- float **temperature**

5.6.1 Detailed Description

Temperature Sensor implementation.

5.6.2 Macro Definition Documentation

5.6.2.1 VCC_TARGET

```
#define VCC_TARGET 3.3
```

Parameters

VCC_TARGET	R21 board VCC voltage used as reference

5.6.3 Function Documentation

5.6.3.1 SET_IO1X_LED_OFF()

```
void SET_IO1X_LED_OFF (
             void  )
```

Turns the IO1X board LED off.

Turn the IO1X expansion board LED off.

5.6.3.2 SET_IO1X_LED_ON()

```
void SET_IO1X_LED_ON (
             void  )
```

Turns the IO1X board LED on.

Turn the IO1X expansion board LED on.

5.6.3.3 UART_write_byte()

```
void UART_write_byte (
            uint8_t byte_to_send )
```

Writes a byte on the debugger UART.

Writes a byte in the debugging UART.

5.7 TempAcele_e_Iluini_main.h

Go to the documentation of this file.

```
00001
00005 #include <atmel_start.h>
00006 #include <stdio.h>
00007 #include <bno055.h>
00008
00012 #define VCC_TARGET 3.3
00013
00014 volatile bool conversion_done = false;
00015 struct io_descriptor* USART_debug_io;
00016 struct bno055_dev bno055;
00017
00022 static void convert_cb_Light_sensor_ADC(const struct adc_async_descriptor *const descr, const uint8_t
      channel)
00023 {
00024     conversion_done = true;
00025 }
00026
00031 void Light_sensor_init(void)
00032 {
00033     adc_async_register_callback(&Light_sensor_ADC, 0, ADC_ASYNC_CONVERT_CB,
      convert_cb_Light_sensor_ADC);
00034     adc_async_enable_channel(&Light_sensor_ADC, 0);
00035     adc_async_start_conversion(&Light_sensor_ADC);
00036 }
00037
00043 void USART_dbg_init(void){
00044     usart_sync_get_io_descriptor(&USART_debug, &USART_debug_io);
00045 }
00046
00051 void UART_write_byte(uint8_t byte_to_send){
00052     io_write(USART_debug_io, &byte_to_send, 1);
00053 }
00054
00060 float GET_light_sensor(void){
00061     uint8_t lightSensorValue;
00062     float voltageSensor;
00063     float iluminance;
00064
00065     // Faz a conversão AD do sensor de luz
00066     adc_async_start_conversion(&Light_sensor_ADC);
00067     while(!conversion_done){}
00068     adc_async_read_channel(&Light_sensor_ADC, 0, &lightSensorValue, 1);
00069
00070     // Faz a definição dos valores de tensão lidos do sensor a partir dos dados quantizados do ADC
00071     voltageSensor = lightSensorValue * VCC_TARGET / 255;
00072
00073     // Calcula a iluminância com base na corrente do resistor de coletor do fototransistor e na
      relação entre lux e corrente
00074     iluminance = (VCC_TARGET - voltageSensor) * 200;
00075
00076     return iluminance;
00077 }
00078
00083 void SET_IO1X_LED_ON(void){
00084     gpio_set_pin_level(LED, true);
00085 }
00086
00091 void SET_IO1X_LED_OFF(void){
00092     gpio_set_pin_level(LED, false);
00093 }
00094
00100 void floatToString(float num, char* str, int precision) {
00101     int i = 0;
00102
00103     // Extract the integral part
```

```
00104     int integralPart = (int)num;
00105
00106     // Convert the integral part to string
00107     do {
00108         str[i++] = integralPart % 10 + '0';
00109         integralPart /= 10;
00110     } while (integralPart > 0);
00111
00112     // Reverse the integral part string
00113     int j;
00114     int len = i;
00115     for (j = 0; j < len / 2; j++) {
00116         char temp = str[j];
00117         str[j] = str[len - j - 1];
00118         str[len - j - 1] = temp;
00119     }
00120
00121     // Add decimal point
00122     str[i++] = '.';
00123
00124     // Extract the fractional part
00125     float fractionalPart = num - (int)num;
00126
00127     // Convert the fractional part to string
00128     int k;
00129     for (k = 0; k < precision; k++) {
00130         fractionalPart *= 10;
00131         int digit = (int)fractionalPart;
00132         str[i++] = digit + '0';
00133         fractionalPart -= digit;
00134     }
00135
00136     // Add null-terminating character
00137     str[i] = '\0';
00138 }
00139
00140 float acceleration_x; // Aceleração no eixo X
00141 float acceleration_y; // Aceleração no eixo Y
00142 float acceleration_z; // Aceleração no eixo Z
00143
00144 float temperature; // Temperatura lida do sensor
00145
00146 int main(void)
00147 {
00148     // Initializes MCU, drivers and middleware
00149     atmel_start_init();
00150     Light_sensor_init();
00151     USART_dbg_init();
00152
00153     char message[50];
00154     char acceleration_str[10];
00155     char temperature_str[10];
00156
00157     // Accel initialization
00158     struct bno055_init(&bno055);
00159
00160     while (1) {
00161         // Reads accel data
00162         struct bno055_read_acceleration(&bno055, &acceleration_x, &acceleration_y, &acceleration_z);
00163
00164         // Reads temperature data
00165         temperature = bno055_get_temperature(&bno055);
00166
00167         // Formats acceleration data into string
00168         floatToString(acceleration_x, acceleration_str, 4);
00169         floatToString(temperature, temperature_str, 2);
00170
00171         // UART Message
00172         sprintf(message, "Aceleracao X: %s\r\nTemperatura: %s\r\n", acceleration_str,
      temperature_str);
00173         printf(message);
00174
00175         // LED On and off
00176         delay_ms(20);
00177         SET_IO1X_LED_OFF();
00178         delay_ms(20);
00179     }
00180 }
```

5.8 ot-rtos_elc1048-main/HAL_Sensores/HAL_Io1_Plained/Test_Io1_Plained/temperature_sensor.c File Reference

Temperature Sensor implementation.

```
#include <temperature_sensor.h>
```

Functions

- struct temperature_sensor * temperature_sensor_construct (struct temperature_sensor *const me, void *const io, const struct temperature_sensor_interface *const interface)

  Construct abstract temperature sensor.
- float temperature_sensor_read (const struct temperature_sensor *const me)

  Read temperature from the given sensor.

5.8.1 Detailed Description

Temperature Sensor implementation.

Copyright (c) 2016-2018 Microchip Technology Inc. and its subsidiaries.

\asf_license_start

5.8.2 Function Documentation

5.8.2.1 temperature_sensor_construct()

```
struct temperature_sensor * temperature_sensor_construct (
            struct temperature_sensor *const me,
            void *const io,
            const struct temperature_sensor_interface *const interface )
```

Construct abstract temperature sensor.

Parameters

in	*me*	The pointer to temperature sensor to initialize
in	*io*	The pointer to instance of interface to actual sensor
in	*interface*	The pointer to interface of temperature sensor

Returns

pointer to initialized sensor

5.8.2.2 temperature_sensor_read()

```
float temperature_sensor_read (
```

```
            const struct temperature_sensor *const me )
```

Read temperature from the given sensor.

Parameters

in	*me*	The pointer to temperature sensor to read temperature from

Returns

temperature

5.9 ot-rtos_elc1048-main/HAL_Sensores/HAL_Io1_Plained/Test_Io1_↩ Plained/temperature_sensor.h File Reference

Temperature Sensor declaration.

```
#include <compiler.h>
```

Data Structures

- struct temperature_sensor_interface
  Interface of abstract temperature sensor.
- struct temperature_sensor
  Abstract temperature sensor.

Functions

- struct temperature_sensor * temperature_sensor_construct (struct temperature_sensor *const me, void *const io, const struct temperature_sensor_interface *const interface)
  Construct abstract temperature sensor.
- float temperature_sensor_read (const struct temperature_sensor *const me)
  Read temperature from the given sensor.

5.9.1 Detailed Description

Temperature Sensor declaration.

Copyright (c) 2016-2018 Microchip Technology Inc. and its subsidiaries.

\asf_license_start

5.9.2 Function Documentation

5.9.2.1 temperature_sensor_construct()

```
struct temperature_sensor * temperature_sensor_construct (
            struct temperature_sensor *const me,
            void *const io,
            const struct temperature_sensor_interface *const interface )
```

Construct abstract temperature sensor.

Parameters

in	*me*	The pointer to temperature sensor to initialize
in	*io*	The pointer to instance of interface to actual sensor
in	*interface*	The pointer to interface of temperature sensor

Returns

pointer to initialized sensor

5.9.2.2 temperature_sensor_read()

```
float temperature_sensor_read (
            const struct temperature_sensor *const me )
```

Read temperature from the given sensor.

Parameters

in	*me*	The pointer to temperature sensor to read temperature from

Returns

temperature

5.10 temperature_sensor.h

Go to the documentation of this file.

```
00001
00034 #ifndef _TEMPERATURE_SENSOR_H_INCLUDED
00035 #define _TEMPERATURE_SENSOR_H_INCLUDED
00036
00037 #include <compiler.h>
00038
00039 #ifdef __cplusplus
00040 extern "C" {
00041 #endif
00042
00044 struct temperature_sensor;
00045
00049 struct temperature_sensor_interface {
00050     float (*read)(const struct temperature_sensor *const me);
00051 };
00052
00056 struct temperature_sensor {
00058     void *io;
00060     const struct temperature_sensor_interface *interface;
00061 };
00062
00072 struct temperature_sensor *temperature_sensor_construct(struct temperature_sensor *const me, void
      *const io,
00073                                                         const struct temperature_sensor_interface
      *const interface);
00074
00082 float temperature_sensor_read(const struct temperature_sensor *const me);
00083
00084 #ifdef __cplusplus
00085 }
00086 #endif
00087
00088 #endif /* _TEMPERATURE_SENSOR_H_INCLUDED */
```

Index