Codice Sistemi Embedded

Generated by Doxygen 1.8.13

Contents

Chapter 1

Documentazione codice sistemi embedded

Table of Contents

1.1 GPIO

1.1.1 Driver

1.1.1.1 UIO

- Funzioni per la gestione del driver GPIO_interrupt_uio_poll.c

1.1.1.2 Kernel

- Modulo kernel che permette di interagire con la periferica GPIO_kernel_main.c
- Permette la gestione di un gruppo di periferiche dello stesso tipo GPIO_list.c
- Funzionalità utilizzate per controllare un singolo dipositivo GPIO.c

1.1.2 Hardware

- Controlla la generazione dell' interrupt GPIO_v1_0_S00_AXI.vhd
- Top level entity del componente GPIO_v1_0_S00_AXI GPIO_v1_0.vhd

1.2 UART

1.2.1 Driver

1.2.1.1 UIO

- gestione del componente UART utilizzando il driver uio UART_interrupt_uio.c

1.2.1.2 KERNEL

- Modulo kernel che permette di interagire con la periferica UART_kernel_main.c
- Permette la gestione di un gruppo di periferiche dello stesso tipo UART_list.c
- Funzionalità utilizzate per controllare un singolo dipositivo UART.c UART_interrupt_kernel_mode.c

1.2.2 Hardware

- Controlla la generazione dell' interrupt UART_v1_0_S00_AXI.vhd
- Top level entity del componente UART_v1_0_S00_AXI UART_v1_0.vhd

1.3 Progetto_finale

- gestione dell' invio e ricazione dei dati sulle varie periferiche con calcolo e check del CRC main.c

1.3.1 Periferiche

1.3.1.1 CAN

- funzioni per configurare la periferica CAN can.c

1.3.1.2 SPI

- funzioni per configurare la periferica SPI spi.c

1.3.1.3 I2C

- funzioni per configurare la periferica I2C i2c.c

1.3.1.4 UART

- funzioni per configurare la periferica UART usart.c

1.3.1.5 GPIO

- funzioni per configurare i banchi del GPIO gpio.c

Chapter 2

driver_UART_UIO

funzioni per gestire la trasmissione e la ricezione dei dati utilizzando il protocollo UART

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Design Unit Index

4.1 Design Unit Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 5

Design Unit Index

5.1 Design Unit List

Here is a list of all design unit members with links to the Entities they belong to:

Chapter 6

File Index

6.1 File List

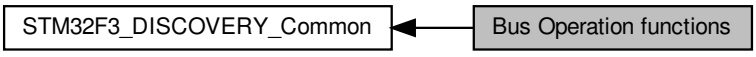
Here is a list of all documented files with brief descriptions:

Chapter 7

Module Documentation

7.1 Bus Operation functions

Collaboration diagram for Bus Operation functions:



7.1.1 Detailed Description

7.1.2 Function Documentation

7.1.2.1 I2Cx_Error()

```
static void I2Cx_Error (
             void  )  [static]
```

I2C3 error treatment function.

Return values

None	

7.1.2.2 I2Cx_Init()

```
static void I2Cx_Init (
             void  )  [static]
```

Discovery I2Cx Bus initialization.

Return values

None	

7.1.2.3 I2Cx_MspInit()

```
static void I2Cx_MspInit (
             I2C_HandleTypeDef * hi2c )  [static]
```

Discovery I2Cx MSP Initialization.

Parameters

hi2c	I2C handle

Return values

None	

7.1.2.4 I2Cx_ReadData()

```
static uint8_t I2Cx_ReadData (
             uint16_t Addr,
             uint8_t Reg )  [static]
```

Read a value in a register of the device through BUS.

Parameters

Addr	Device address on BUS Bus.
Reg	The target register address to write

Return values

Data	read at register @

7.1.2.5 I2Cx_WriteData()

```
static void I2Cx_WriteData (
             uint16_t Addr,
             uint8_t Reg,
             uint8_t Value )  [static]
```

Write a value in a register of the device through BUS.

Parameters

Addr	Device address on BUS Bus.
Reg	The target register address to write
Value	The target register value to be written

Return values

None	

7.1.2.6 SPIx_Error()

```
static void SPIx_Error (
             void  )  [static]
```

SPIx error treatment function.

Return values

None	

7.1.2.7 SPIx_Init()

```
static void SPIx_Init (
             void  )  [static]
```

SPIx Bus initialization.

Return values

None	

7.1.2.8 SPIx_MspInit()

```
static void SPIx_MspInit (
            SPI_HandleTypeDef * hspi )  [static]
```

SPI MSP Init.

Parameters

hspi	SPI handle

Return values

None	

7.1.2.9 SPIx_WriteRead()

```
static uint8_t SPIx_WriteRead (
            uint8_t Byte )  [static]
```

Sends a Byte through the SPI interface and return the Byte received from the SPI bus.

Parameters

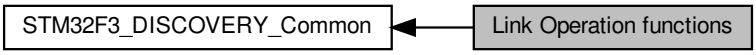
Byte	Byte send.

Return values

The	received byte value

7.2 Link Operation functions

Collaboration diagram for Link Operation functions:



7.2.1 Detailed Description

7.2.2 Function Documentation

7.2.2.1 COMPASSACCELERO_IO_Init()

```
void COMPASSACCELERO_IO_Init (
             void  )
```

Configures COMPASS / ACCELEROMETER I2C interface.

Return values

None	

7.2.2.2 COMPASSACCELERO_IO_ITConfig()

```
void COMPASSACCELERO_IO_ITConfig (
             void  )
```

Configures COMPASS / ACCELERO click IT.

Return values

None	

7.2.2.3 COMPASSACCELERO_IO_Read()

```
uint8_t COMPASSACCELERO_IO_Read (
```

```
            uint16_t DeviceAddr,
            uint8_t RegisterAddr )
```

Reads a block of data from the COMPASS / ACCELEROMETER.

Parameters

DeviceAddr	specifies the slave address to be programmed(ACC_I2C_ADDRESS or MAG_I2C_ADDRESS).
RegisterAddr	specifies the COMPASS / ACCELEROMETER internal address register to read from

Return values

ACCELEROMETER	register value

7.2.2.4 COMPASSACCELERO_IO_Write()

```
void COMPASSACCELERO_IO_Write (
            uint16_t DeviceAddr,
            uint8_t RegisterAddr,
            uint8_t Value )
```

Writes one byte to the COMPASS / ACCELEROMETER.

Parameters

DeviceAddr	specifies the slave address to be programmed.
RegisterAddr	specifies the COMPASS / ACCELEROMETER register to be written.
Value	Data to be written

Return values

None	

7.2.2.5 GYRO_IO_Init()

```
void GYRO_IO_Init (
            void  )
```

Configures the GYROSCOPE SPI interface.

Return values

None	

7.2.2.6 GYRO_IO_Read()

```
void GYRO_IO_Read (
            uint8_t* pBuffer,
            uint8_t ReadAddr,
            uint16_t NumByteToRead )
```

Reads a block of data from the GYROSCOPE.

Parameters

pBuffer	pointer to the buffer that receives the data read from the GYROSCOPE.
ReadAddr	GYROSCOPE's internal address to read from.
NumByteToRead	number of bytes to read from the GYROSCOPE.

Return values

None	

7.2.2.7 GYRO_IO_Write()

```
void GYRO_IO_Write (
            uint8_t* pBuffer,
            uint8_t WriteAddr,
            uint16_t NumByteToWrite )
```

Writes one byte to the GYROSCOPE.

Parameters

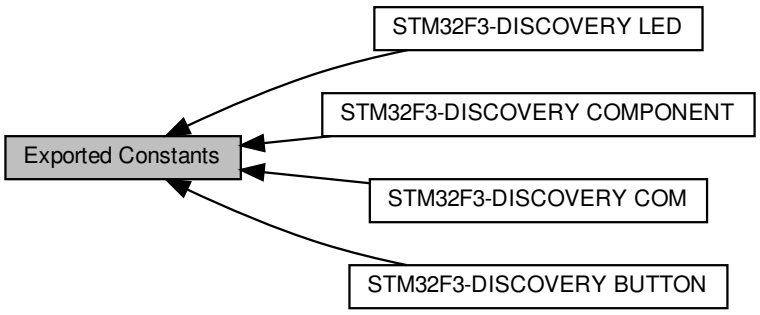
pBuffer	pointer to the buffer containing the data to be written to the GYROSCOPE.
WriteAddr	GYROSCOPE's internal address to write to.
NumByteToWrite	Number of bytes to write.

Return values

None	

7.3 Exported Constants

Collaboration diagram for Exported Constants:



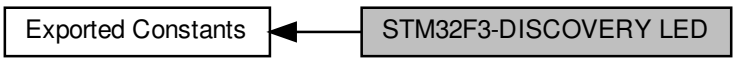
Modules

- STM32F3-DISCOVERY LED
- STM32F3-DISCOVERY BUTTON
- STM32F3-DISCOVERY COM
- STM32F3-DISCOVERY COMPONENT

7.3.1 Detailed Description

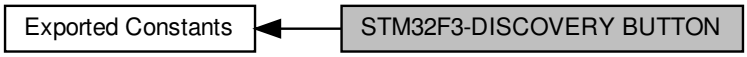
7.4 STM32F3-DISCOVERY LED

Collaboration diagram for STM32F3-DISCOVERY LED:



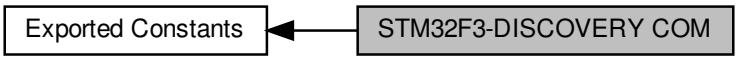
7.5 STM32F3-DISCOVERY BUTTON

Collaboration diagram for STM32F3-DISCOVERY BUTTON:



7.6 STM32F3-DISCOVERY COM

Collaboration diagram for STM32F3-DISCOVERY COM:

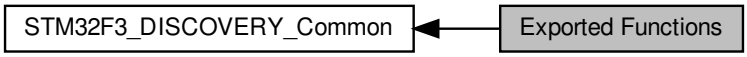


7.7 STM32F3-DISCOVERY COMPONENT

Collaboration diagram for STM32F3-DISCOVERY COMPONENT:

7.8 Exported Functions

Collaboration diagram for Exported Functions:



7.8.1 Detailed Description

7.8.2 Function Documentation

7.8.2.1 BSP_GetVersion()

```
uint32_t BSP_GetVersion (
             void  )
```

This method returns the STM32F3-DISCOVERY BSP Driver revision.

Return values

version	: 0xXYZR (8bits for each decimal, R for RC)

7.8.2.2 BSP_LED_Init()

```
void BSP_LED_Init (
             Led_TypeDef Led )
```

Configures LED GPIO.

Parameters

Led	Specifies the Led to be configured. This parameter can be one of following parameters: • LED_RED • LED_BLUE • LED_ORANGE • LED_GREEN • LED_GREEN2 • LED_ORANGE2 • LED_BLUE2 • LED_RED2

Return values

None	

7.8.2.3 BSP_LED_Off()

```
void BSP_LED_Off (
             Led_TypeDef Led )
```

Turns selected LED Off.

Parameters

Led	Specifies the Led to be set off. This parameter can be one of following parameters: • LED_RED • LED_BLUE • LED_ORANGE • LED_GREEN • LED_GREEN2 • LED_ORANGE2 • LED_BLUE2 • LED_RED2

Return values

None	

7.8.2.4 BSP_LED_On()

```
void BSP_LED_On (
            Led_TypeDef Led )
```

Turns selected LED On.

Parameters

Led	Specifies the Led to be set on. This parameter can be one of following parameters: • LED_RED • LED4 • LED5 • LED6 • LED7 • LED8 • LED9 • LED10

Return values

None	

7.8.2.5 BSP_LED_Toggle()

```
void BSP_LED_Toggle (
            Led_TypeDef Led )
```

Toggles the selected LED.

Parameters

Led	Specifies the Led to be toggled. This parameter can be one of following parameters: • LED_RED • LED_BLUE • LED_ORANGE • LED_GREEN • LED_GREEN2 • LED_ORANGE2 • LED_BLUE2 • LED_RED2

Return values

None	

7.8.2.6 BSP_PB_GetState()

```
uint32_t BSP_PB_GetState (
            Button_TypeDef Button )
```

Returns the selected Push Button state.

Parameters

Button	Specifies the Button to be checked. This parameter should be: BUTTON_USER

Return values

The	Button GPIO pin value.

7.8.2.7 BSP_PB_Init()

```
void BSP_PB_Init (
            Button_TypeDef Button,
            ButtonMode_TypeDef ButtonMode )
```

Configures Push Button GPIO and EXTI Line.

Parameters

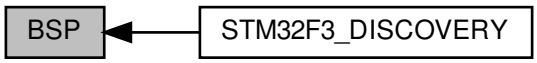
Button	Specifies the Button to be configured. This parameter should be: BUTTON_USER
ButtonMode	Specifies Button mode. This parameter can be one of following parameters: • BUTTON_MODE_GPIO: Button will be used as simple IO • BUTTON_MODE_EXTI: Button will be connected to EXTI line with interrupt generation capability

Return values

None	

7.9 BSP

Collaboration diagram for BSP:



Modules

- STM32F3_DISCOVERY

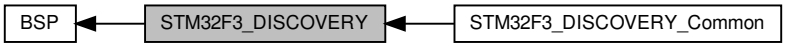
  This file provides set of firmware functions to manage Leds and push-button available on STM32F3-Discovery Kit from STMicroelectronics.

7.9.1 Detailed Description

7.10 STM32F3_DISCOVERY

This file provides set of firmware functions to manage Leds and push-button available on STM32F3-Discovery Kit from STMicroelectronics.

Collaboration diagram for STM32F3_DISCOVERY:



Modules

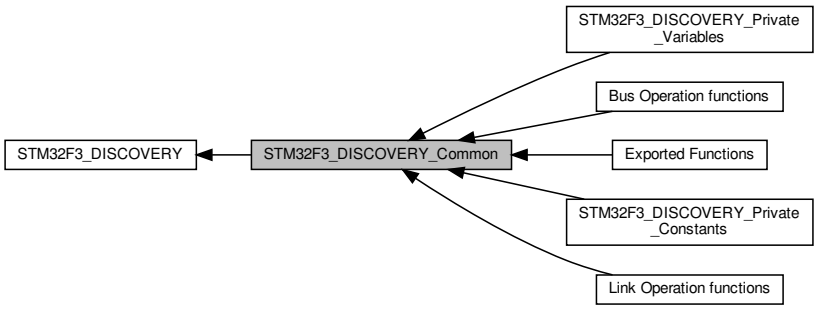
- STM32F3_DISCOVERY_Common

7.10.1 Detailed Description

This file provides set of firmware functions to manage Leds and push-button available on STM32F3-Discovery Kit from STMicroelectronics.

7.11 STM32F3_DISCOVERY_Common

Collaboration diagram for STM32F3_DISCOVERY_Common:



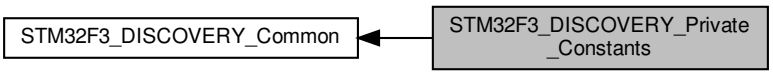
Modules

- Bus Operation functions
- Link Operation functions
- STM32F3_DISCOVERY_Private_Constants
- STM32F3_DISCOVERY_Private_Variables
- Exported Functions

7.11.1 Detailed Description

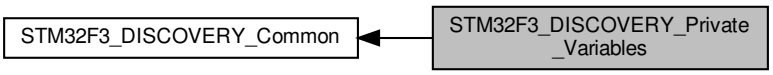
7.12 STM32F3_DISCOVERY_Private_Constants

Collaboration diagram for STM32F3_DISCOVERY_Private_Constants:



7.13 STM32F3_DISCOVERY_Private_Variables

Collaboration diagram for STM32F3_DISCOVERY_Private_Variables:



7.13.1 Detailed Description

7.13.2 Variable Documentation

7.13.2.1 LED_PIN

```
const uint16_t LED_PIN[LEDn]
```

Initial value:

```
= {LED3_PIN, LED4_PIN, LED5_PIN, LED6_PIN,
                                LED7_PIN, LED8_PIN, LED9_PIN, LED10_PIN}
```

7.13.2.2 LED_PORT

```
GPIO_TypeDef* LED_PORT[LEDn]
```

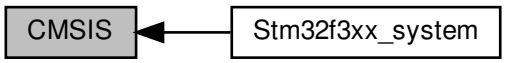
Initial value:

```
= {LED3_GPIO_PORT, LED4_GPIO_PORT, LED5_GPIO_PORT, LED6_GPIO_PORT,
                                LED7_GPIO_PORT, LED8_GPIO_PORT, LED9_GPIO_PORT, LED10_GPIO_PORT}
```

LED variables.

7.14 CMSIS

Collaboration diagram for CMSIS:



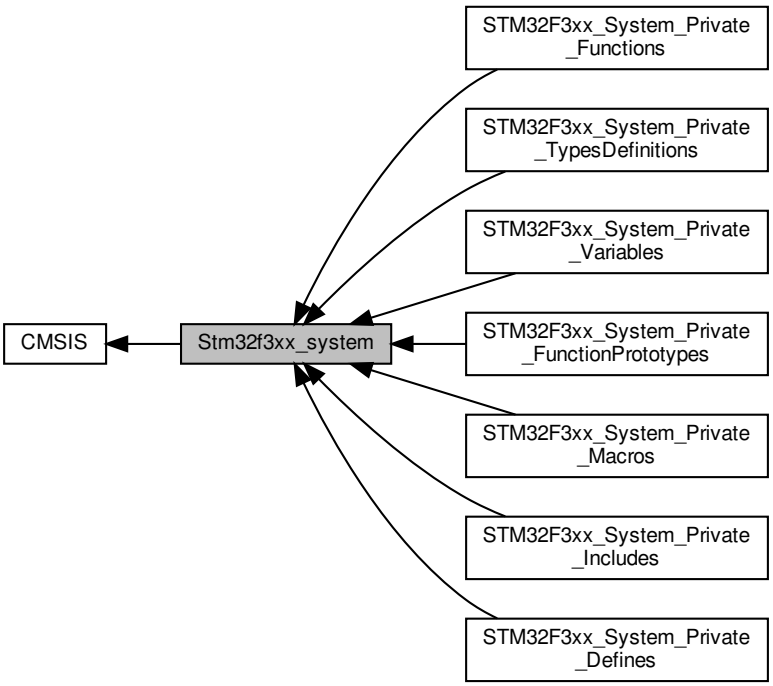
Modules

- Stm32f3xx_system

7.14.1 Detailed Description

7.15 Stm32f3xx_system

Collaboration diagram for Stm32f3xx_system:



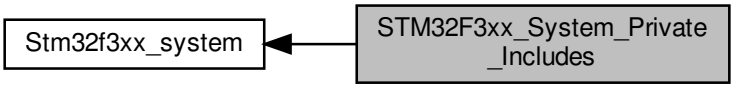
Modules

- STM32F3xx_System_Private_Includes
- STM32F3xx_System_Private_TypesDefinitions
- STM32F3xx_System_Private_Defines
- STM32F3xx_System_Private_Macros
- STM32F3xx_System_Private_Variables
- STM32F3xx_System_Private_FunctionPrototypes
- STM32F3xx_System_Private_Functions

7.15.1 Detailed Description

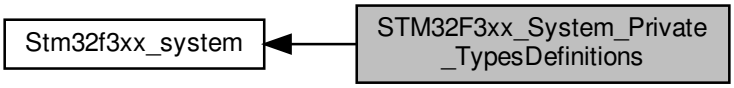
7.16 STM32F3xx_System_Private_Includes

Collaboration diagram for STM32F3xx_System_Private_Includes:



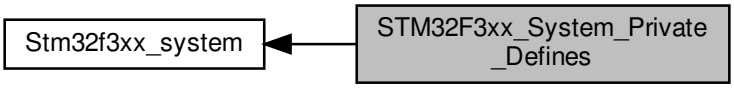
7.17 STM32F3xx_System_Private_TypesDefinitions

Collaboration diagram for STM32F3xx_System_Private_TypesDefinitions:



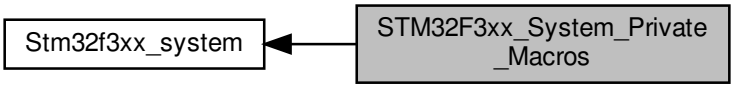
7.18 STM32F3xx_System_Private_Defines

Collaboration diagram for STM32F3xx_System_Private_Defines:



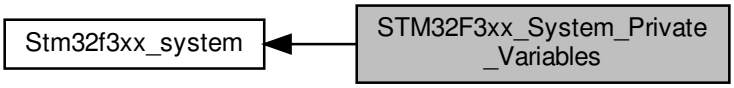
7.19 STM32F3xx_System_Private_Macros

Collaboration diagram for STM32F3xx_System_Private_Macros:



7.20 STM32F3xx_System_Private_Variables

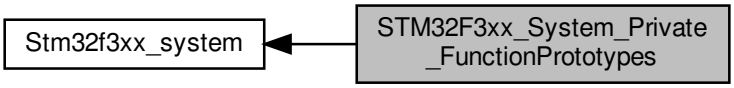
Collaboration diagram for STM32F3xx_System_Private_Variables:



7.20.1 Detailed Description

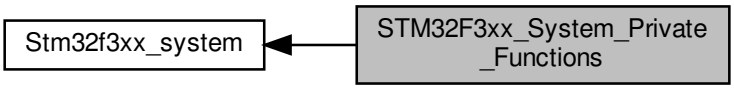
7.21 STM32F3xx_System_Private_FunctionPrototypes

Collaboration diagram for STM32F3xx_System_Private_FunctionPrototypes:



7.22 STM32F3xx_System_Private_Functions

Collaboration diagram for STM32F3xx_System_Private_Functions:



7.22.1 Detailed Description

7.22.2 Function Documentation

7.22.2.1 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 stm32f3xx_hal.h file (default value 8 MHz) but the real value may vary depending on the variations in voltage and temperature.

(∗∗) HSE_VALUE is a constant defined in stm32f3xx_hal.h file (default value 8 MHz), 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	

7.22.2.2 SystemInit()

```
void SystemInit (
             void  )
```

Setup the microcontroller system Initialize the FPU setting, vector table location and the PLL configuration is reset.

Parameters

None	

Return values

None	

Chapter 8

Data Structure Documentation

8.1 arch_imp Architecture Reference

Components

- GPIO_v1_0_S00_AXI

Instantiations

- gpio_v1_0_s00_axi_inst **GPIO_v1_0_S00_AXI**

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

- /media/saverio/OS/Users/Saverio/Desktop/SE/git/codici_da_mandare/FPGA/GPIO/Hardware/GPIO_1.↩ 0/hdl/GPIO_v1_0.vhd

8.2 arch_imp Architecture Reference

Processes

- PROCESS_0(**S_AXI_ACLK**)
- PROCESS_1(**S_AXI_ACLK**)
- PROCESS_2(**S_AXI_ACLK**)
- PROCESS_3(**S_AXI_ACLK**)
- PROCESS_4(**S_AXI_ACLK**)
- PROCESS_5(**S_AXI_ACLK**)
- PROCESS_6(**S_AXI_ACLK**)
- PROCESS_7(**slv_reg0 , slv_reg1 , gpio_read , slv_reg3 , slv_reg4 , slv_reg5 , status_reg_out , slv_**↩ **reg7_out , axi_araddr , S_AXI_ARESETN , slv_reg_rden**)
- PROCESS_8(**S_AXI_ACLK**)
- gpio_read_sampling(**S_AXI_ACLK , gpio_read**)

  Campiona i segnali di cui si vuole verificare la generazione di un interrupt.
- intr_pending(**S_AXI_ACLK , change_detected , ack_intr**)

  Gestisce il registro pending.
- inst_irq(**S_AXI_ACLK , pending_intr**)

Components

- GPIO_Array

Constants

- ADDR_LSB **integer:=(C_S_AXI_DATA_WIDTH/ 32)+ 1**
- OPT_MEM_ADDR_BITS **integer:= 2**

Signals

- axi_awaddr **std_logic_vector(C_S_AXI_ADDR_WIDTH- 1 downto 0)**
- axi_awready **std_logic**
- axi_wready **std_logic**
- axi_bresp **std_logic_vector(1 downto 0)**
- axi_bvalid **std_logic**
- axi_araddr **std_logic_vector(C_S_AXI_ADDR_WIDTH- 1 downto 0)**
- axi_arready **std_logic**
- axi_rdata **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- axi_rresp **std_logic_vector(1 downto 0)**
- axi_rvalid **std_logic**
- slv_reg0 **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- slv_reg1 **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- slv_reg2 **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- slv_reg3 **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- slv_reg4 **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- slv_reg5 **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- slv_reg6 **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- slv_reg7 **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- slv_reg7_out **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- slv_reg_rden **std_logic**
- slv_reg_wren **std_logic**
- reg_data_out **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- byte_index **integer**
- aw_en **std_logic**
- gpio_read **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- status_reg_out **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- status_reg **std_logic_vector(width - 1 downto 0)**

  determina se un segnale ha generato interrupt
- status_reg_tmp **std_logic_vector(width - 1 downto 0)**
- detected_intr **std_logic_vector(width - 1 downto 0)**
- pending_intr **std_logic_vector(width - 1 downto 0)**

  determina se una interruzione è pendente
- pending_intr_tmp **std_logic_vector(width - 1 downto 0)**

  salva interruzioni pendenti precedenti
- changed_bits **std_logic_vector(width - 1 downto 0)**

  identifica quale pin del GPIO sono abilitati descrive se è stato campionanto un interrupt
- last_stage **std_logic_vector(width - 1 downto 0)**

  registro primario del campionatore
- current_stage **std_logic_vector(width - 1 downto 0)**

  registro secondario del campionatore
- change_detected **std_logic**

  determina se è avvenuta una condizione che ha generato un interrupt

Instantiations

- inst_gpio_array **gpio_array**

Aliases

- global_intr **std_logicisslv_reg3(0)**
- intr_mask **std_logic_vector(width - 1 downto 0)isslv_reg4(width - 1 downto 0)**

  determina se le interruzioni globali sono attive
- ack_intr **std_logic_vector(width - 1 downto 0)isslv_reg7(width - 1 downto 0)**

  determina quali interrupt sono attivi
- gpio_enable **std_logic_vector(width - 1 downto 0)isslv_reg0(width - 1 downto 0)**

  determina quali segnali di interrupt pendenti sono stati catturati dal driver

8.2.1 Member Function Documentation

8.2.1.1 gpio_read_sampling()

```
gpio_read_sampling (
              S_AXI_ACLK,
              gpio_read )
```

Campiona i segnali di cui si vuole verificare la generazione di un interrupt.

Parameters

in	*S_AXI_ACLK*	clock del bus AXI
in	*gpio_read*	valori del GPIO da campionare

8.2.1.2 inst_irq()

```
inst_irq(
              S_AXI_ACLK  ,
              pending_intr   )  [Process]
```

Per la descrizione del componente riferirsi alla documentazione dell' intero design

8.2.1.3 intr_pending()

```
intr_pending(
              S_AXI_ACLK  ,
              change_detected   ,
              ack_intr   )  [Process]
```

Gestisce il registro pending.

Per la descrizione del componente riferirsi alla documentazione dell' intero design

Parameters

in	*S_AXI_ACLK*	clock del bus AXI
in	*change_detected*	identifica l' avvenimento dell' interruput su un segnale abilitato
in	*ack_intr*	cattura un segnale di ack generato dal driver che gestisce l' eccezione

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

- /media/saverio/OS/Users/Saverio/Desktop/SE/git/codici_da_mandare/FPGA/GPIO/Hardware/GPIO_1.↩ 0/hdl/GPIO_v1_0_S00_AXI.vhd

8.3 arch_imp Architecture Reference

Processes

- PROCESS_9**(S_AXI_ACLK)**

  dato ricevuto
- PROCESS_10**(S_AXI_ACLK)**

  segnale il cui valore alto indica che un nuovo dato ricevuto è dispobile
- PROCESS_11**(S_AXI_ACLK)**
- PROCESS_12**(S_AXI_ACLK)**
- PROCESS_13**(S_AXI_ACLK)**
- PROCESS_14**(S_AXI_ACLK)**
- PROCESS_15**(S_AXI_ACLK)**
- PROCESS_16**(slv_reg0 , slv_reg1 , uart_status_reg , slv_reg3_out , slv_reg4 , slv_reg5 , slv_reg6 , slv_reg7_out , axi_araddr , S_AXI_ARESETN , slv_reg_rden)**
- PROCESS_17**(S_AXI_ACLK)**
- status_reg_sampling**(S_AXI_ACLK , uart_status_reg)**

  Campiona i segnali di cui si vuole verificare la generazione di un interrupt.
- intr_pending**(S_AXI_ACLK , change_detected , ack_intr , pending_intr_tmp , changed_bits)**

  Gestisce il registro pending.
- inst_irq**(S_AXI_ACLK , pending_intr , global_intr)**

  Disabilita l' interrupt nel caso di reset del bus e tiene alto il segnale di interrupt finchè rimane pendente.

Components

- UART

  UART.

Constants

- ADDR_LSB **integer:=(C_S_AXI_DATA_WIDTH/** 32 **)+** 1
- OPT_MEM_ADDR_BITS **integer:=** 2

Signals

- axi_awaddr **std_logic_vector(C_S_AXI_ADDR_WIDTH- 1 downto 0)**
- axi_awready **std_logic**
- axi_wready **std_logic**
- axi_bresp **std_logic_vector(1 downto 0)**
- axi_bvalid **std_logic**
- axi_araddr **std_logic_vector(C_S_AXI_ADDR_WIDTH- 1 downto 0)**
- axi_arready **std_logic**
- axi_rdata **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- axi_rresp **std_logic_vector(1 downto 0)**
- axi_rvalid **std_logic**
- slv_reg0 **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- slv_reg1 **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- slv_reg2 **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- slv_reg3 **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- slv_reg4 **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- slv_reg5 **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- slv_reg6 **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0):=(others=>' 0 ')**
- slv_reg7 **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- slv_reg_rden **std_logic**
- slv_reg_wren **std_logic**
- reg_data_out **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- byte_index **integer**
- aw_en **std_logic**
- uart_status_reg **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- slv_reg3_out **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- slv_reg7_out **std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- reset **std_logic**
- pending_intr **std_logic_vector(1 downto 0)**

  interruzioni pendenti
- pending_intr_tmp **std_logic_vector(1 downto 0)**

  delay intr_pending
- changed_bits **std_logic_vector(1 downto 0)**
- tx_busy_falling_detect **std_logic**

  vale 1 quando viene rilevato il falling_edge di tx_busy
- rx_rising_detect **std_logic**

  alto quando viene rilevato il rising_edge di RDA
- last_stage **std_logic_vector(1 downto 0)**
- current_stage **std_logic_vector(1 downto 0)**
- change_detected **std_logic**

Instantiations

- inst_uart **uart**

Aliases

- global_intr **std_logicisslv_reg4(0)**
- intr_mask **std_logic_vector(1 downto 0)isslv_reg5(1 downto 0)**

  enable interruzioni IP CORE
- ack_intr **std_logic_vector(1 downto 0)isslv_reg7(1 downto 0)**

8.3.1 Member Function Documentation

8.3.1.1 inst_irq()

```
inst_irq(
              S_AXI_ACLK  ,
              pending_intr   ,
              global_intr  )  [Process]
```

Disabilita l' interrupt nel caso di reset del bus e tiene alto il segnale di interrupt finchè rimane pendente.

Per la descrizione del componente riferirsi alla documentazione dell' intero design

Parameters

in	*S_AXI_ACLK*	clock del bus AXI
in	*pending_intr*	registro che identifica le interruzioni pendenti

8.3.1.2 intr_pending()

```
intr_pending(
              S_AXI_ACLK  ,
              change_detected  ,
              ack_intr   ,
              pending_intr_tmp   ,
              changed_bits   )  [Process]
```

Gestisce il registro pending.

Per la descrizione del componente riferirsi alla documentazione dell' intero design

Parameters

in	*S_AXI_ACLK*	clock del bus AXI
in	*change_detected*	identifica l' avvenimento dell' interruput su un segnale abilitato
in	*ack_intr*	cattura un segnale di ack generato dal driver che gestisce l' eccezione

8.3.1.3 status_reg_sampling()

```
status_reg_sampling (
              S_AXI_ACLK,
              uart_status_reg )
```

Campiona i segnali di cui si vuole verificare la generazione di un interrupt.

Parameters

in	*S_AXI_ACLK*	clock del bus AXI
in	*uart_status_reg*	valori del UART da campionare

8.3.2 Field Documentation

8.3.2.1 ack_intr

```
ack_intr std_logic_vector( 1 downto 0 )isslv_reg7( 1 downto 0 )   [Alias]
```

maschera interruzioni rda(1) e tx_busy(0). Mettendo il relativo bit ad uno si abilita la lina di interruzione

8.3.2.2 changed_bits

```
changed_bits std_logic_vector( 1 downto 0 )   [Signal]
```

segnale di ack. Il bit 0 da ack all'interuzione della trasmissione, il bit 1 a quello dela ricezione. Logica 1 attiva

8.3.2.3 UART

```
UART    [Component]
```

UART.

componente contenente un ricevitore e un trasmettitore che implementano il protocollo UART. Consulatare documentazione esterna.

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

- /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/FPGA/ip_repo/UART_1.0/hdl/UART_v1_0_S00↩_AXI.vhd

8.4 arch_imp Architecture Reference

componente UART_AXI_S00 componente nel quale è incapsulato il componente UART e la logica di gestione delle interruzioni.

Components

- UART_v1_0_S00_AXI

Instantiations

- uart_v1_0_s00_axi_inst **UART_v1_0_S00_AXI**

8.4.1 Detailed Description

componente UART_AXI_S00 componente nel quale è incapsulato il componente UART e la logica di gestione delle interruzioni.

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

- /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/FPGA/ip_repo/UART_1.0/hdl/UART_v1_0.vhd

8.5 GPIO Struct Reference

Stuttura che astrae un device GPIO in kernel-mode. Contiene ciò che è necessario al funzionamento del driver.

`#include <GPIO.h>`

Data Fields

8.5.1 Detailed Description

Stuttura che astrae un device GPIO in kernel-mode. Contiene ciò che è necessario al funzionamento del driver.

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

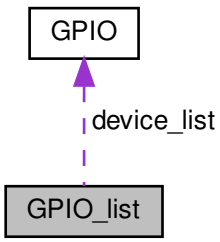
- /media/saverio/OS/Users/Saverio/Desktop/SE/git/codici_da_mandare/FPGA/GPIO/Driver/Kernel_Mode/GPIO.h

8.6 GPIO_list Struct Reference

Struttura dati per la gestione di più device GPIO da parte del driver.

`#include <GPIO_list.h>`

Collaboration diagram for GPIO_list:



Data Fields

8.6.1 Detailed Description

Struttura dati per la gestione di più device GPIO da parte del driver.

8.6.2 Field Documentation

8.6.2.1 device_count

```
uint32_t device_count
```

numero di device attivi e gestiti dal driver

8.6.2.2 device_list

```
GPIO** device_list
```

array di puntatori a GPIO, ciascuno dei quali si riferisce ad un device

8.6.2.3 list_size

```
uint32_t list_size
```

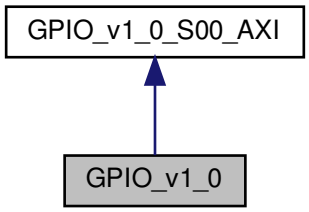
dimensione della lista, ovvero il numero massimo di device gestibili

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

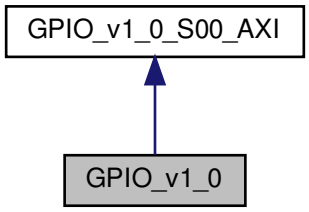
- /media/saverio/OS/Users/Saverio/Desktop/SE/git/codici_da_mandare/FPGA/GPIO/Driver/Kernel_Mode/GPIO_list.h

8.7 GPIO_v1_0 Entity Reference

Inheritance diagram for GPIO_v1_0:



Collaboration diagram for GPIO_v1_0:



Entities

- arch_imp architecture

Libraries

- ieee

  Viene utilizzata la libreria IEEE.

Use Clauses

- std_logic_1164

  Sono utilizzati i segnali della standard logic.
- numeric_std

  Vengono utilizzate le funzioni numeriche.

Generics

- width **integer:= 4**

  determina il numero di GPIO da controllare
- C_S00_AXI_DATA_WIDTH **integer:= 32**
- C_S00_AXI_ADDR_WIDTH **integer:= 5**

Ports

- pads **inout std_logic_vector(width - 1 downto 0)**

  se GPIO in modalità lettura mostra il valore letto, altrimenti forza un valore in uscita
- interrupt **out std_logic**

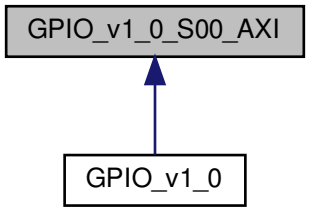
  segnale di interrupt
- s00_axi_aclk **in std_logic**
- s00_axi_aresetn **in std_logic**
- s00_axi_awaddr **in std_logic_vector(C_S00_AXI_ADDR_WIDTH- 1 downto 0)**
- s00_axi_awprot **in std_logic_vector(2 downto 0)**
- s00_axi_awvalid **in std_logic**
- s00_axi_awready **out std_logic**
- s00_axi_wdata **in std_logic_vector(C_S00_AXI_DATA_WIDTH- 1 downto 0)**
- s00_axi_wstrb **in std_logic_vector((C_S00_AXI_DATA_WIDTH/ 8)- 1 downto 0)**
- s00_axi_wvalid **in std_logic**
- s00_axi_wready **out std_logic**
- s00_axi_bresp **out std_logic_vector(1 downto 0)**
- s00_axi_bvalid **out std_logic**
- s00_axi_bready **in std_logic**
- s00_axi_araddr **in std_logic_vector(C_S00_AXI_ADDR_WIDTH- 1 downto 0)**
- s00_axi_arprot **in std_logic_vector(2 downto 0)**
- s00_axi_arvalid **in std_logic**
- s00_axi_arready **out std_logic**
- s00_axi_rdata **out std_logic_vector(C_S00_AXI_DATA_WIDTH- 1 downto 0)**
- s00_axi_rresp **out std_logic_vector(1 downto 0)**
- s00_axi_rvalid **out std_logic**
- s00_axi_rready **in std_logic**

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

- /media/saverio/OS/Users/Saverio/Desktop/SE/git/codici_da_mandare/FPGA/GPIO/Hardware/GPIO_1.↩
  0/hdl/GPIO_v1_0.vhd

8.8 GPIO_v1_0_S00_AXI Entity Reference

Inheritance diagram for GPIO_v1_0_S00_AXI:



Entities

- arch_imp architecture

Libraries

- ieee

  Viene utilizzato la libreria IEEE.

Use Clauses

- std_logic_1164

  Sono utilizzati i segnali della standard logic.
- numeric_std

  Vengono utilizzate le funzioni numeriche.
- std_logic_misc

  Viene utilizzata la libreria misc di utility.

Generics

- width **integer:=** 4

  determina il numero di GPIO da controllare
- C_S_AXI_DATA_WIDTH **integer:=** 32
- C_S_AXI_ADDR_WIDTH **integer:=** 5

Ports

- pads **inout std_logic_vector(width - 1 downto 0)**

  se GPIO in modalità lettura mostra il valore letto, altrimenti forza un valore in uscita
- interrupt **out std_logic**

  segnale di interrupt
- S_AXI_ACLK **in std_logic**
- S_AXI_ARESETN **in std_logic**
- S_AXI_AWADDR **in std_logic_vector(C_S_AXI_ADDR_WIDTH- 1 downto 0)**
- S_AXI_AWPROT **in std_logic_vector(2 downto 0)**
- S_AXI_AWVALID **in std_logic**
- S_AXI_AWREADY **out std_logic**
- S_AXI_WDATA **in std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- S_AXI_WSTRB **in std_logic_vector((C_S_AXI_DATA_WIDTH/ 8)- 1 downto 0)**
- S_AXI_WVALID **in std_logic**
- S_AXI_WREADY **out std_logic**
- S_AXI_BRESP **out std_logic_vector(1 downto 0)**
- S_AXI_BVALID **out std_logic**
- S_AXI_BREADY **in std_logic**
- S_AXI_ARADDR **in std_logic_vector(C_S_AXI_ADDR_WIDTH- 1 downto 0)**
- S_AXI_ARPROT **in std_logic_vector(2 downto 0)**
- S_AXI_ARVALID **in std_logic**
- S_AXI_ARREADY **out std_logic**
- S_AXI_RDATA **out std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- S_AXI_RRESP **out std_logic_vector(1 downto 0)**
- S_AXI_RVALID **out std_logic**
- S_AXI_RREADY **in std_logic**

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

- /media/saverio/OS/Users/Saverio/Desktop/SE/git/codici_da_mandare/FPGA/GPIO/Hardware/GPIO_1.↩ 0/hdl/GPIO_v1_0_S00_AXI.vhd

8.9 UART_list Struct Reference

Struttura dati per la gestione di più device UART da parte del driver.

```
#include <UART_list.h>
```

Data Fields

8.9.1 Detailed Description

Struttura dati per la gestione di più device UART da parte del driver.

8.9.2 Field Documentation

8.9.2.1 device_count

```
uint32_t device_count
```

numero di device attivi e gestiti dal driver

8.9.2.2 device_list

```
UART** device_list
```

array di puntatori a UART, ciascuno dei quali si riferisce ad un device

8.9.2.3 list_size

```
uint32_t list_size
```

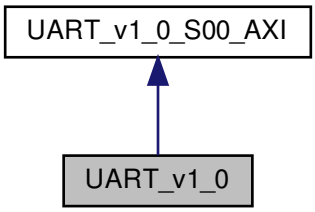
dimensione della lista, ovvero il numero massimo di device gestibili

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

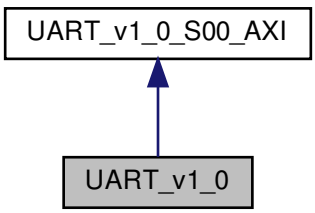
- /media/saverio/OS/Users/Saverio/Desktop/SE/git/Michele/FPGA/UART/Driver/Con_interrupt/KERNEL_M↩ ODE/UART_list.h

8.10 UART_v1_0 Entity Reference

Inheritance diagram for UART_v1_0:



Collaboration diagram for UART_v1_0:



Entities

- arch_imp architecture

  componente UART_AXI_S00 componente nel quale è incapsulato il componente UART e la logica di gestione delle interruzioni.

Libraries

- ieee

  Viene utilizzata la libreria IEEE.

Use Clauses

- std_logic_1164

  Sono utilizzati i segnali della standard logic.
- numeric_std

  Vengono utilizzate le funzioni numeriche.

Generics

- baudrate **integer:=** 9600

  baudare trasmissione
- clock_freq **integer:=** 50_000_000

  frequenza clock ingresso
- C_S00_AXI_DATA_WIDTH **integer:=** 32
- C_S00_AXI_ADDR_WIDTH **integer:=** 5

Ports

- tx **out std_logic**

  linea uscita per la trasmissione
- rx **in std_logic**

  linea ingresso per la ricezione
- interrupt **out std_logic**

  segnale per richiede l'interrupt
- s00_axi_aclk **in std_logic**
- s00_axi_aresetn **in std_logic**
- s00_axi_awaddr **in std_logic_vector(C_S00_AXI_ADDR_WIDTH- 1 downto 0)**
- s00_axi_awprot **in std_logic_vector(2 downto 0)**
- s00_axi_awvalid **in std_logic**
- s00_axi_awready **out std_logic**
- s00_axi_wdata **in std_logic_vector(C_S00_AXI_DATA_WIDTH- 1 downto 0)**
- s00_axi_wstrb **in std_logic_vector((C_S00_AXI_DATA_WIDTH/ 8)- 1 downto 0)**
- s00_axi_wvalid **in std_logic**
- s00_axi_wready **out std_logic**
- s00_axi_bresp **out std_logic_vector(1 downto 0)**
- s00_axi_bvalid **out std_logic**
- s00_axi_bready **in std_logic**
- s00_axi_araddr **in std_logic_vector(C_S00_AXI_ADDR_WIDTH- 1 downto 0)**

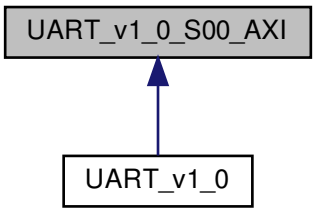
- s00_axi_arprot **in std_logic_vector(2 downto 0)**
- s00_axi_arvalid **in std_logic**
- s00_axi_arready **out std_logic**
- s00_axi_rdata **out std_logic_vector(C_S00_AXI_DATA_WIDTH- 1 downto 0)**
- s00_axi_rresp **out std_logic_vector(1 downto 0)**
- s00_axi_rvalid **out std_logic**
- s00_axi_rready **in std_logic**

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

- /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/FPGA/ip_repo/UART_1.0/hdl/UART_v1_0.vhd

8.11 UART_v1_0_S00_AXI Entity Reference

Inheritance diagram for UART_v1_0_S00_AXI:



Entities

- arch_imp architecture

Libraries

- ieee

  Viene utilizzata la libreria IEEE.

Use Clauses

- std_logic_1164

  Sono utilizzati i segnali della standard logic.
- numeric_std

  Vengono utilizzate le funzioni numeriche.
- std_logic_misc

  libreria necessaria per la funzione or_reduce

Generics

- baudrate **integer:=** 9600
- clock_freq **integer:=** 50_000_000
- C_S_AXI_DATA_WIDTH **integer:=** 32
- C_S_AXI_ADDR_WIDTH **integer:=** 5

Ports

- tx **out std_logic**
- rx **in std_logic**
- interrupt **out std_logic**
- S_AXI_ACLK **in std_logic**
- S_AXI_ARESETN **in std_logic**
- S_AXI_AWADDR **in std_logic_vector(C_S_AXI_ADDR_WIDTH- 1 downto 0)**
- S_AXI_AWPROT **in std_logic_vector(2 downto 0)**
- S_AXI_AWVALID **in std_logic**
- S_AXI_AWREADY **out std_logic**
- S_AXI_WDATA **in std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- S_AXI_WSTRB **in std_logic_vector((C_S_AXI_DATA_WIDTH/ 8)- 1 downto 0)**
- S_AXI_WVALID **in std_logic**
- S_AXI_WREADY **out std_logic**
- S_AXI_BRESP **out std_logic_vector(1 downto 0)**
- S_AXI_BVALID **out std_logic**
- S_AXI_BREADY **in std_logic**
- S_AXI_ARADDR **in std_logic_vector(C_S_AXI_ADDR_WIDTH- 1 downto 0)**
- S_AXI_ARPROT **in std_logic_vector(2 downto 0)**
- S_AXI_ARVALID **in std_logic**
- S_AXI_ARREADY **out std_logic**
- S_AXI_RDATA **out std_logic_vector(C_S_AXI_DATA_WIDTH- 1 downto 0)**
- S_AXI_RRESP **out std_logic_vector(1 downto 0)**
- S_AXI_RVALID **out std_logic**
- S_AXI_RREADY **in std_logic**

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

- /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/FPGA/ip_repo/UART_1.0/hdl/UART_v1_0_S00_AXI.vhd

Chapter 9

File Documentation

9.1 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/FPGA/ip_repo/UART_↩ 1.0/hdl/UART_v1_0.vhd File Reference

UART AXI IPCORE with interrupt.

Entities

- UART_v1_0 entity
- arch_imp architecture

  componente UART_AXI_S00 componente nel quale è incapsulato il componente UART e la logica di gestione delle interruzioni.

9.1.1 Detailed Description

UART AXI IPCORE with interrupt.

9.2 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/FPGA/ip_repo/UART_↩ 1.0/hdl/UART_v1_0_S00_AXI.vhd File Reference

UART AXI IPCORE with interrupt.

Entities

- UART_v1_0_S00_AXI entity
- arch_imp architecture

9.2.1 Detailed Description

UART AXI IPCORE with interrupt.

9.3 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_↩ CHECK/Inc/can.h File Reference

9.3.1 Detailed Description

header file per la configurazione della periferica CAN

9.3.2 Function Documentation

9.3.2.1 MX_CAN_Init()

```
void MX_CAN_Init (
            uint16_t nodeAddress,
            uint16_t groupAddress )
```

Funzione di configurazione della periferica CAN modalità di utilizzo, filtri.

Parameters

nodeAddress	setta l' indentificativo del nodo
groupAddress	setta l' identificato del gruppo a cui il nodo appartiene

9.4 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_↩ CHECK/Inc/crc.h File Reference

9.4.1 Detailed Description

header file per la configurazione della periferica CRC

9.4.2 Function Documentation

9.4.2.1 MX_CRC_Init()

```
void MX_CRC_Init (
            uint32_t CRC_Polynomial,
            uint32_t CRC_DefaultValue )
```

Funzione di configurazione della periferica CRC.

Parameters

CRC_Polynomial	polinomio utilizzato per calcolare il CRC
CRC_DefaultValue	valore utilizzato per effettura una operazione di XOR prima che il CRC venga calcolato a cui il nodo appartiene

9.5 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_↩ CHECK/Inc/gpio.h File Reference

9.5.1 Detailed Description

header file per la configurazione dei banchi di GPIO

9.5.2 Function Documentation

9.5.2.1 LedOff()

```
void LedOff ( )
```

Spegne tutti i led che sono utilizzati nel codice.

Parameters

9.5.2.2 MX_GPIO_Init()

```
void MX_GPIO_Init (
             void  )
```

Funzione di configurazione dei vari banchi di GPIO.

Parameters

9.6 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_↩ CHECK/Inc/i2c.h File Reference

9.6.1 Detailed Description

header file per la configurazione della periferica I2C

9.6.2 Function Documentation

9.6.2.1 MX_I2C2_Init()

```
void MX_I2C2_Init (
            uint16_t nodeAddress,
            uint16_t groupAddress )
```

Funzione di configurazione della periferica I2C.

Parameters

nodeAddress	setta l' indentificativo del nodo
groupAddress	setta l' identificato del gruppo a cui il nodo appartiene

9.7 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_↩ CHECK/Inc/main.h File Reference

: Header for main.c file. This file contains the common defines of the application.

9.7.1 Detailed Description

: Header for main.c file. This file contains the common defines of the application.

Attention

9.8 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_↩ CHECK/Inc/spi.h File Reference

9.8.1 Detailed Description

header file per la configurazione della periferica SPI

9.8.2 Function Documentation

9.8.2.1 MX_SPI2_Init()

```
void MX_SPI2_Init (
            void  )
```

Funzione di configurazione della periferica SPI.

Parameters

9.9 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_↩ CHECK/Inc/stm32f3_discovery.h File Reference

This file contains definitions for STM32F3-Discovery's Leds, push- buttons hardware resources.

9.9.1 Detailed Description

This file contains definitions for STM32F3-Discovery's Leds, push- buttons hardware resources.

Author

MCD Application Team

Attention

9.10 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_↩ CHECK/Inc/stm32f3xx_hal_conf.h File Reference

HAL configuration file.

9.10.1 Detailed Description

HAL configuration file.

Attention

9.10.2 Variable Documentation

9.10.2.1 C

C

Value of the External oscillator in Hz Time out for HSE start up, in ms Value of the Internal oscillator in Hz Time out for HSI start up 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 Time out for LSE start up, in ms Value of the External oscillator in Hz Value of VDD in mv tick interrupt priority (lowest by default)

9.11 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_↩ CHECK/Inc/stm32f3xx_it.h File Reference

This file contains the headers of the interrupt handlers.

9.11.1 Detailed Description

This file contains the headers of the interrupt handlers.

Attention

9.11.2 Variable Documentation

9.11.2.1 C

C

Initial value:

```
{
#endif
```

```
void NMI_Handler(void)
```

9.12 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_↩ CHECK/Inc/usart.h File Reference

9.12.1 Detailed Description

header file per la configurazione della periferica USART

9.12.2 Function Documentation

9.12.2.1 MX_USART2_UART_Init()

```
void MX_USART2_UART_Init (
            uint32_t Baudrate )
```

Funzione di configurazione della periferica USART.

Parameters

Baudrate	setta il baudrate della periferica

9.13 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_↩ CHECK/Src/can.c File Reference

9.13.1 Detailed Description

Permette la configurazione della periferica CAN

9.13.2 Function Documentation

9.13.2.1 HAL_CAN_MspDeInit()

```
void HAL_CAN_MspDeInit (
            CAN_HandleTypeDef* canHandle )
```

Disabilita la periferica CAN.

Parameters

canHandle	handler della periferica CAN

9.13.2.2 HAL_CAN_MspInit()

```
void HAL_CAN_MspInit (
            CAN_HandleTypeDef* canHandle )
```

Configura opportunamente l' handler della periferica CAN ed i pin associati ad essa.

Parameters

canHandle	handler della periferica CAN

9.13.2.3 MX_CAN_Init()

```
void MX_CAN_Init (
```

```
            uint16_t nodeAddress,
            uint16_t groupAddress )
```

Funzione di configurazione della periferica CAN modalità di utilizzo, filtri.

Parameters

nodeAddress	setta l' indentificativo del nodo
groupAddress	setta l' identificato del gruppo a cui il nodo appartiene

9.14 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_↩ CHECK/Src/crc.c File Reference

9.14.1 Detailed Description

Permetta la configurazione della periferica CRC

9.14.2 Function Documentation

9.14.2.1 HAL_CRC_MspDeInit()

```
void HAL_CRC_MspDeInit (
            CRC_HandleTypeDef* crcHandle )
```

Disabilita la periferica CRC.

Parameters

crcHandle	handler della periferica CRC

9.14.2.2 HAL_CRC_MspInit()

```
void HAL_CRC_MspInit (
            CRC_HandleTypeDef* crcHandle )
```

Configura opportunamente l' handler della periferica CRC ed i pin associati ad essa.

Parameters

crcHandle	handler della periferica CRC

9.14.2.3 MX_CRC_Init()

```
void MX_CRC_Init (
            uint32_t CRC_Polynomial,
            uint32_t CRC_DefaultValue )
```

Funzione di configurazione della periferica CRC.

Parameters

CRC_Polynomial	polinomio utilizzato per calcolare il CRC
CRC_DefaultValue	valore utilizzato per effettura una operazione di XOR prima che il CRC venga calcolato a cui il nodo appartiene

9.15 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_CHECK/Src/gpio.c File Reference

9.15.1 Detailed Description

Configura i banchi di GPIO

9.15.2 Function Documentation

9.15.2.1 LedOff()

```
void LedOff ( )
```

Spegne tutti i led che sono utilizzati nel codice.

Parameters

9.15.2.2 MX_GPIO_Init()

```
void MX_GPIO_Init (
            void  )
```

Funzione di configurazione dei vari banchi di GPIO.

Parameters

9.16 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_↩ CHECK/Src/i2c.c File Reference

9.16.1 Detailed Description

Permette la configurazione della periferica I2C

9.16.2 Function Documentation

9.16.2.1 HAL_I2C_MspDeInit()

```
void HAL_I2C_MspDeInit (
            I2C_HandleTypeDef* i2cHandle )
```

Disabilita la periferica CAN.

Parameters

canHandle	handler della periferica CAN

9.16.2.2 HAL_I2C_MspInit()

```
void HAL_I2C_MspInit (
            I2C_HandleTypeDef* i2cHandle )
```

Configura opportunamente l' handler della periferica I2C ed i pin associati ad essa.

Parameters

i2cHandle	handler della periferica I2C

9.16.2.3 MX_I2C2_Init()

```
void MX_I2C2_Init (
```

```
            uint16_t nodeAddress,
            uint16_t groupAddress )
```

Funzione di configurazione della periferica I2C.

Parameters

nodeAddress	setta l' indentificativo del nodo
groupAddress	setta l' identificato del gruppo a cui il nodo appartiene

9.17 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_CHECK/Src/main.c File Reference

programma main che permette a due board di comunicare utilizzando diversi dispositivi di input ed output

9.17.1 Detailed Description

programma main che permette a due board di comunicare utilizzando diversi dispositivi di input ed output

9.17.2 Function Documentation

9.17.2.1 Configure_Peripheral()

```
void Configure_Peripheral (
            uint8_t peripheral,
            uint16_t nodeAddress,
            uint16_t groupAddress )
```

Configura le periferiche affinchè possano ricevere ed inviare messaggi.

Parameters

peripheral	valore che indica quale periferiche abilitare
nodeAddress	indirizzo del nodo da contattare, utilizzato se la comunicazione che lo prevede
groupAddress	indirizzo del gruppo da contattare, utilizzato se la comunicazione che lo prevede

9.17.2.2 CRC_Check()

```
void CRC_Check (
            uint32_t * ReceivedFrame )
```

Controlla che i due CRC ricevuti siano corretti.

Parameters

ReceivedFrame	frame ricevuto

9.17.2.3 Frame32to8()

```
void Frame32to8 (
            uint32_t * in_buffer32,
            uint8_t * out_buffer8 )
```

Converte un frame da un formato uint32_t ad uno uint8_t.

Parameters

in_buffer32	puntatore ad un dato di tipo uint32_t
out_buffer8	puntatore ad un dato di tipo uint8_t

9.17.2.4 Frame8to32()

```
void Frame8to32 (
            uint8_t * in_buffer8,
            uint32_t * out_buffer32 )
```

Converte un frame da un formato uint8_t ad uno uint32_t.

Parameters

in_buffer8	puntatore ad un dato di tipo uint8_t
out_buffer32	puntatore ad un dato di tipo uint32_t

9.17.2.5 getSSPin()

```
uint16_t getSSPin (
            uint16_t address )
```

ritorna il pin GPIO a cui è collegato lo slave dato l' indirizzo

Parameters

address	indirizzo della periferica SPI

9.17.2.6 HAL_CAN_RxFifo0MsgPendingCallback()

```
MX void HAL_CAN_RxFifo0MsgPendingCallback (
            CAN_HandleTypeDef * hcan )
```

Viene utilizzata per sapere se ci sono messaggi da leggere pendenti nel buffer fifo di ricezione del CAN.

Parameters

hcan	handler alla struttura che gestisce CAN

9.17.2.7 HAL_CAN_TxMailbox0CompleteCallback()

```
void HAL_CAN_TxMailbox0CompleteCallback (
            CAN_HandleTypeDef * hcan )
```

Indica che tutti i messaggi che dovevano essere mandati da quella memoria sono stati inviati.

Parameters

hcan	handler alla struttura che gestisce CAN

9.17.2.8 HAL_GPIO_EXTI_Callback()

```
void HAL_GPIO_EXTI_Callback (
            uint16_t GPIO_Pin )
```

Viene utilizzata per sapere quando viene premuto l' user button.

Parameters

GPIO_Pin	il pin del GPIO a cui è collegato il pin

9.17.2.9 HAL_I2C_ErrorCallback()

```
void HAL_I2C_ErrorCallback (
            I2C_HandleTypeDef * hi2c2 )
```

Funzione chiamata nel caso di errore di comunicazione sul canale I2C.

Parameters

hi2c2	handler alla struttura che gestisce I2C

Return values

None	

1- When Slave don't acknowledge it's address, Master restarts communication. 2- When Master don't acknowledge the last data transferred, Slave don't care in this example.

9.17.2.10 HAL_I2C_MasterRxCpltCallback()

```
void HAL_I2C_MasterRxCpltCallback (
            I2C_HandleTypeDef * hi2c2 )
```

Funzione chiamata alla ricezione di dati sul canale I2C da parte di uno master.

Parameters

hi2c2	handler alla struttura che gestisce I2C

9.17.2.11 HAL_I2C_MasterTxCpltCallback()

```
void HAL_I2C_MasterTxCpltCallback (
            I2C_HandleTypeDef * hi2c2 )
```

Funzione chiamata dopo l' invio di dati sul canale I2C da parte di un master.

Parameters

hi2c2	handler alla struttura che gestisce I2C

9.17.2.12 HAL_I2C_SlaveRxCpltCallback()

```
void HAL_I2C_SlaveRxCpltCallback (
            I2C_HandleTypeDef * hi2c2 )
```

Funzione chiamata alla ricezione di dati sul canale I2C da parte di uno slave.

Parameters

hi2c2	handler alla struttura che gestisce I2C

9.17.2.13 HAL_I2C_SlaveTxCpltCallback()

```
void HAL_I2C_SlaveTxCpltCallback (
            I2C_HandleTypeDef * hi2c2 )
```

Funzione chiamata dopo l' invio di dati sul canale I2C da parte di uno slave.

Parameters

hi2c2	handler alla struttura che gestisce I2C

9.17.2.14 HAL_SPI_ErrorCallback()

```
void HAL_SPI_ErrorCallback (
            SPI_HandleTypeDef * hspi )
```

Funzione chiamata nel caso di errore di comunicazione sul canale SPI.

Parameters

hspi	handler alla struttura che gestisce SPI

9.17.2.15 HAL_SPI_RxCpltCallback()

```
void HAL_SPI_RxCpltCallback (
            SPI_HandleTypeDef * hspi )
```

Funzione chiamata alla ricezione di dati sul canale SPI.

Parameters

hspi	handler alla struttura che gestisce SPI

9.17.2.16 HAL_SPI_TxCpltCallback()

```
void HAL_SPI_TxCpltCallback (
            SPI_HandleTypeDef * hspi )
```

Funzione chiamata dopo l' invio di dati sul canale SPI.

Parameters

hspi	handler alla struttura che gestisce SPI

9.17.2.17 HAL_UART_ErrorCallback()

```
void HAL_UART_ErrorCallback (
            UART_HandleTypeDef * UartHandle )
```

Funzione chiamata nel caso di errore di comunicazione sul canale UART.

Parameters

UartHandle	handler alla struttura che gestisce UART

9.17.2.18 HAL_UART_RxCpltCallback()

```
void HAL_UART_RxCpltCallback (
            UART_HandleTypeDef * UartHandle )
```

Funzione chiamata alla ricezione di dati sul canale UART.

Parameters

UartHandle	handler alla struttura che gestisce UART

9.17.2.19 HAL_UART_TxCpltCallback()

```
void HAL_UART_TxCpltCallback (
            UART_HandleTypeDef * UartHandle )
```

Funzione chiamata dopo l' invio di dati sul canale UART.

Parameters

UartHandle	handler alla struttura che gestisce UART

9.17.2.20 Receive_CRC()

```
uint8_t Receive_CRC (
            uint32_t * ReceivedData,
            uint8_t channel,
            uint16_t address )
```

Abilita la ricezione del frame sulle differenti periferiche.

Parameters

ReceivedData	struttura contenete i dati ricevuti
channel	indica le periferiche da cui voglio ricevere
address	indica lo slave SPI con cui voglio comunicare, permettendomi di scegliere lo slave select opportuno

9.17.2.21 Send_CRC()

```
void Send_CRC (
            uint32_t * MSG,
            uint16_t address,
            uint8_t channel )
```

Invia il messaggio sulle varie periferiche.

Parameters

MSG	messaggio da inviare
address	indirizzo della periferica da contattare se previsto dalla modalità di comunicazione
channel	indica le periferiche su cui voglio inviare

9.17.2.22 SystemClock_Config()

```
void SystemClock_Config (
            void  )
```

Gestisce il clock di sistema.

Initializes the CPU, AHB and APB busses clocks

Initializes the CPU, AHB and APB busses clocks

9.18 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_↩ CHECK/Src/spi.c File Reference

9.18.1 Detailed Description

Permette la configurazione della periferica SPI

9.18.2 Function Documentation

9.18.2.1 HAL_SPI_MspDeInit()

```
void HAL_SPI_MspDeInit (
            SPI_HandleTypeDef* spiHandle )
```

Disabilita la periferica SPI.

Parameters

spiHandle	handler della periferica SPI

9.18.2.2 HAL_SPI_MspInit()

```
void HAL_SPI_MspInit (
            SPI_HandleTypeDef* spiHandle )
```

Configura opportunamente l' handler della periferica SPI ed i pin associati ad essa.

Parameters

spiHandle	handler della periferica SPI

9.18.2.3 MX_SPI2_Init()

```
void MX_SPI2_Init (
            void  )
```

Funzione di configurazione della periferica SPI.

Parameters

9.19 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_↩ CHECK/Src/stm32f3_discovery.c File Reference

This file provides set of firmware functions to manage Leds and push-button available on STM32F3-DISCOVERY Kit from STMicroelectronics.

9.19.1 Detailed Description

This file provides set of firmware functions to manage Leds and push-button available on STM32F3-DISCOVERY Kit from STMicroelectronics.

Author

MCD Application Team

Attention

9.20 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_↩ CHECK/Src/stm32f3xx_it.c File Reference

Interrupt Service Routines.

9.20.1 Detailed Description

Interrupt Service Routines.

Attention

9.21 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_↩ CHECK/Src/system_stm32f3xx.c File Reference

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

9.21.1 Detailed Description

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

Author

MCD Application Team

1. 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_stm32f3xx.s" file.
   - SystemCoreClock variable: Contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.
   - SystemCoreClockUpdate(): Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.
2. After each device reset the HSI (8 MHz) is used as system clock source. Then SystemInit() function is called, in "startup_stm32f3xx.s" file, to configure the system clock before to branch to main program.

9.21.2 3. This file configures the system clock as follows:

9.21.2.1 Supported STM32F3xx device

9.21.2.2 System Clock source | HSI

9.21.2.3 SYSCLK(Hz) | 8000000

9.21.2.4 HCLK(Hz) | 8000000

9.21.2.5 AHB Prescaler | 1

9.21.2.6 APB2 Prescaler | 1

9.21.2.7 APB1 Prescaler | 1

9.21.2.8 USB Clock | DISABLE

==========

Attention

9.22 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Andrea/STM32/CRC_MultiSerial_↩ CHECK/Src/usart.c File Reference

9.22.1 Detailed Description

Permette la configurazione della periferica USART

9.22.2 Function Documentation

9.22.2.1 HAL_UART_MspDeInit()

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

Disabilita la periferica UASRT.

Parameters

uartHandle	handler della periferica USART

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

9.22.2.2 HAL_UART_MspInit()

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

Configura opportunamente l' handler della periferica USART ed i pin associati ad essa.

Parameters

uartHandle	handler della periferica USART

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

9.22.2.3 MX_USART2_UART_Init()

```
void MX_USART2_UART_Init (
            uint32_t Baudrate )
```

Funzione di configurazione della periferica USART.

Parameters

Baudrate	setta il baudrate della periferica

9.23 /media/saverio/OS/Users/Saverio/Desktop/SE/git/codici_da_mandare/FPGA/GPIO/↩ Driver/Kernel_Mode/GPIO.c File Reference

Funzioni utilizzate per interagire con la singola entità GPIO permette la gestione dell' interrupt.

9.23.1 Detailed Description

Funzioni utilizzate per interagire con la singola entità GPIO permette la gestione dell' interrupt.

9.23.2 Function Documentation

9.23.2.1 GPIO_Destroy()

```
void GPIO_Destroy (
            GPIO* device )
```

Rimuove un device GPIO con le relative strutture kernel allocate per il suo funzionamento.

Parameters

device	puntatore a struttura GPIO che indica l'istanza GPIO da rimuovere

9.23.2.2 GPIO_GetDeviceAddress()

```
void* GPIO_GetDeviceAddress (
            GPIO* device )
```

Restituisce l'indirizzo virtuale di memoria cui è mappato un device.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare

9.23.2.3 GPIO_GetPollMask()

```
unsigned GPIO_GetPollMask (
            GPIO * device,
            struct file * file_ptr,
            struct poll_table_struct * wait )
```

Verifica che le operazioni di lettura risultino non-bloccanti.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare
file	puntatore al descrittore file del device
wait	puntatore alla struttura poll_table

Returns

maschera di bit che indica se sia possibile effettuare operazioni di lettura non bloccanti.

Back-end di tre diverse sys-calls: poll, epoll e select,

9.23.2.4 GPIO_GlobalInterruptDisable()

```
void GPIO_GlobalInterruptDisable (
            GPIO* device )
```

Disabilitazione interrupt globali;.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare

9.23.2.5 GPIO_GlobalInterruptEnable()

```
void GPIO_GlobalInterruptEnable (
            GPIO* device )
```

Abilitazione interrupt globali;.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare

9.23.2.6 GPIO_Init()

```
int GPIO_Init (
            GPIO* GPIO_device,
            struct module * owner,
            struct platform_device * pdev,
            struct class* class,
            const char* driver_name,
            const char* device_name,
            uint32_t serial,
            struct file_operations * f_ops,
            irq_handler_t irq_handler,
            uint32_t irq_mask )
```

Inizializza una struttura GPIO per il corrispondente device.

Parameters

GPIO_device	puntatore a struttura GPIO, corrispondente al device su cui operare
owner	puntatore a struttura struct module, proprietario del device (THIS_MODULE)
pdev	puntatore a struct platform_device
driver_name	nome del driver
device_name	nome del device
serial	numero seriale del device
f_ops	puntatore a struttura struct file_operations, specifica le funzioni che agiscono sul device
irq_handler	puntatore irq_handler_t alla funzione che gestisce gli interrupt generati dal device
irq_mask	maschera delle interruzioni attive del device

Return values

0	se non si è verificato nessun errore

Inizializzazione della wait-queue per la system-call read() e poll()

Inizializzazione degli spinlock

Abilitazione degli interrupt del device

9.23.2.7 GPIO_PendingPinInterrupt()

```
unsigned GPIO_PendingPinInterrupt (
            GPIO* device )
```

Fornisce una maschera che indica quali interrupt non sono ancora stati serviti e che quindi risultano pending.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare

Returns

maschera riportante i pin per i quali gli interrupt non sono stati ancora serviti

9.23.2.8 GPIO_PinInterruptAck()

```
void GPIO_PinInterruptAck (
            GPIO* device,
            unsigned mask )
```

Invia al device notifica di servizio di un interrupt;.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare
mask	maschera di selezione degli interrupt da notificare

9.23.2.9 GPIO_PinInterruptDisable()

```
void GPIO_PinInterruptDisable (
            GPIO* device,
            unsigned mask )
```

Disabilitazione interrupt per i singoli pin del device.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare
mask	maschera di selezione degli interrupt da disabilitare

9.23.2.10 GPIO_PinInterruptEnable()

```
void GPIO_PinInterruptEnable (
            GPIO* device,
            unsigned mask )
```

Abilitazione interrupt per i singoli pin del device.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare
mask	maschera di selezione degli interrupt da abilitare

9.23.2.11 GPIO_ResetCanRead()

```
void GPIO_ResetCanRead (
            GPIO* device )
```

Utilizzata per resettare il flag "can_read" di uno specifico device GPIO.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare

9.23.2.12 GPIO_SetCanRead()

```
void GPIO_SetCanRead (
            GPIO* device )
```

Utilizzata per asserire il flag "can_read" di uno specifico device GPIO.

Parameters

device	puntatore a struttura GPIO, device su cui operare

9.23.2.13 GPIO_TestCanReadAndSleep()

```
void GPIO_TestCanReadAndSleep (
            GPIO* device )
```

Testa il valore del flag "can_read". Se è uguale a 0, ovvero non è possibile effettuare una lettura, mette in sleep il processo.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare

9.23.2.14 GPIO_WakeUp()

```
void GPIO_WakeUp (
            GPIO* device )
```

Risveglia i processi in attesa sulle code di read e poll.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare

9.24 /media/saverio/OS/Users/Saverio/Desktop/SE/git/codici_da_mandare/FPGA/GPIO/↩ Driver/Kernel_Mode/GPIO.h File Reference

header file GPIO

Data Structures

- struct GPIO

  Stuttura che astrae un device GPIO in kernel-mode. Contiene ciò che è necessario al funzionamento del driver.

9.24.1 Detailed Description

header file GPIO

9.24.2 Function Documentation

9.24.2.1 GPIO_Destroy()

```
void GPIO_Destroy (
            GPIO* device )
```

Rimuove un device GPIO con le relative strutture kernel allocate per il suo funzionamento.

Parameters

device	puntatore a struttura GPIO che indica l'istanza GPIO da rimuovere

9.24.2.2 GPIO_GetDeviceAddress()

```
void* GPIO_GetDeviceAddress (
            GPIO* device )
```

Restituisce l'indirizzo virtuale di memoria cui è mappato un device.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare

9.24.2.3 GPIO_GetPollMask()

```
unsigned GPIO_GetPollMask (
            GPIO * device,
            struct file * file_ptr,
            struct poll_table_struct * wait )
```

Verifica che le operazioni di lettura risultino non-bloccanti.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare
file	puntatore al descrittore file del device
wait	puntatore alla struttura poll_table

Returns

maschera di bit che indica se sia possibile effettuare operazioni di lettura non bloccanti.

Back-end di tre diverse sys-calls: poll, epoll e select,

9.24.2.4 GPIO_GlobalInterruptDisable()

```
void GPIO_GlobalInterruptDisable (
            GPIO* device )
```

Disabilitazione interrupt globali;.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare

9.24.2.5 GPIO_GlobalInterruptEnable()

```
void GPIO_GlobalInterruptEnable (
            GPIO* device )
```

Abilitazione interrupt globali;.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare

9.24.2.6 GPIO_Init()

```
int GPIO_Init (
            GPIO* GPIO_device,
            struct module * owner,
            struct platform_device * pdev,
            struct class* class,
            const char* driver_name,
            const char* device_name,
            uint32_t serial,
            struct file_operations * f_ops,
            irq_handler_t irq_handler,
            uint32_t irq_mask )
```

Inizializza una struttura GPIO per il corrispondente device.

Parameters

GPIO_device	puntatore a struttura GPIO, corrispondente al device su cui operare
owner	puntatore a struttura struct module, proprietario del device (THIS_MODULE)
pdev	puntatore a struct platform_device
driver_name	nome del driver
device_name	nome del device
serial	numero seriale del device
f_ops	puntatore a struttura struct file_operations, specifica le funzioni che agiscono sul device
irq_handler	puntatore irq_handler_t alla funzione che gestisce gli interrupt generati dal device
irq_mask	maschera delle interruzioni attive del device

Return values

0	se non si è verificato nessun errore

Inizializzazione della wait-queue per la system-call read() e poll()

Inizializzazione degli spinlock

Abilitazione degli interrupt del device

9.24.2.7 GPIO_PendingPinInterrupt()

```
unsigned GPIO_PendingPinInterrupt (
            GPIO* device )
```

Fornisce una maschera che indica quali interrupt non sono ancora stati serviti e che quindi risultano pending.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare

Returns

maschera riportante i pin per i quali gli interrupt non sono stati ancora serviti

9.24.2.8 GPIO_PinInterruptAck()

```
void GPIO_PinInterruptAck (
            GPIO* device,
            unsigned mask )
```

Invia al device notifica di servizio di un interrupt;.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare
mask	maschera di selezione degli interrupt da notificare

9.24.2.9 GPIO_PinInterruptDisable()

```
void GPIO_PinInterruptDisable (
            GPIO* device,
            unsigned mask )
```

Disabilitazione interrupt per i singoli pin del device.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare
mask	maschera di selezione degli interrupt da disabilitare

9.24.2.10 GPIO_PinInterruptEnable()

```
void GPIO_PinInterruptEnable (
            GPIO* device,
            unsigned mask )
```

Abilitazione interrupt per i singoli pin del device.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare
mask	maschera di selezione degli interrupt da abilitare

9.24.2.11 GPIO_ResetCanRead()

```
void GPIO_ResetCanRead (
            GPIO* device )
```

Utilizzata per resettare il flag "can_read" di uno specifico device GPIO.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare

9.24.2.12 GPIO_SetCanRead()

```
void GPIO_SetCanRead (
            GPIO* device )
```

Utilizzata per asserire il flag "can_read" di uno specifico device GPIO.

Parameters

device	puntatore a struttura GPIO, device su cui operare

9.24.2.13 GPIO_TestCanReadAndSleep()

```
void GPIO_TestCanReadAndSleep (
            GPIO* device )
```

Testa il valore del flag "can_read". Se è uguale a 0, ovvero non è possibile effettuare una lettura, mette in sleep il processo.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare

9.24.2.14 GPIO_WakeUp()

```
void GPIO_WakeUp (
            GPIO* device )
```

Risveglia i processi in attesa sulle code di read e poll.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare

9.25 /media/saverio/OS/Users/Saverio/Desktop/SE/git/codici_da_mandare/FPGA/GPIO/↩ Driver/Kernel_Mode/GPIO_kernel_main.c File Reference

Inizializza il driver kernel ed espone le funzionalità del modulo.

9.25.1 Detailed Description

Inizializza il driver kernel ed espone le funzionalità del modulo.

9.25.2 Function Documentation

9.25.2.1 GPIO_irq_handler()

```
static irqreturn_t GPIO_irq_handler (
            int irq,
            struct pt_regs * regs )  [static]
```

Interrupt-handler.

Parameters

irq	Interrupt-number a cui il device è connesso
regs	registri sullo stack alla system call entry

Return values

IRQ_HANDLED	dopo aver servito l'interruzione

9.25.2.2 GPIO_llseek()

```
static loff_t GPIO_llseek (
            struct file * file_ptr,
            loff_t off,
            int whence )  [static]
```

Implementa le system-call lseek() e llseek().

Parameters

file_ptr	puntatore al descrittore file del device
off	offset da aggiungere al parametro whence per il posizionamento
whence	può assumere i valori SEEK_SET, SEEK_CUR o SEEK_END per specificare rispettivamente il riferimento dall'inizio file, dalla posizione corrente o dalla fine.

Returns

Nuova posizione della "testina" di lettura/scrittura

9.25.2.3 GPIO_open()

```
static int GPIO_open (
            struct inode * inode,
            struct file * file_ptr )  [static]
```

Invocata all'apertura del file corrispondente al device.

Return values

0	se non si verifica nessun errore

9.25.2.4 GPIO_poll()

```
static unsigned int GPIO_poll (
            struct file * file_ptr,
            struct poll_table_struct * wait )  [static]
```

Verifica che le operazioni di lettura risultino non-bloccanti.

Parameters

device	puntatore a struttura GPIO, che si riferisce al device su cui operare
file_ptr	puntatore al descrittore file del device
wait	puntatore alla struttura poll_table

Returns

maschera di bit che indica se sia possibile effettuare operazioni di lettura non bloccanti.

Back-end di tre diverse sys-calls: poll, epoll e select,

9.25.2.5 GPIO_probe()

```
static int GPIO_probe (
            struct platform_device * pdev )  [static]
```

Inizializzazione del driver

9.25.2.6 GPIO_read()

```
static ssize_t GPIO_read (
            struct file * file_ptr,
            char * buf,
            size_t count,
            loff_t * off )  [static]
```

Legge dati dal device.

Parameters

file_ptr	puntatore al descrittore file del device
buf	puntatore all'area di memoria dove verranno copiati i count bytes letti
count	numeri di bytes da trasferire
off	long offset type che indica la posizione alla quale si sta effettuando l'accesso

Note

l'aggiunta del flag O_NONBLOCK all'apertura del file descriptor associato al device farà sì che il processo chiamante non verrà bloccato se alla chiamata di una lettura non troverà dati disponibili

9.25.2.7 GPIO_release()

```
static int GPIO_release (
            struct inode * inode,
            struct file * file_ptr )  [static]
```

Invocata alla chiusura del file corrispondente al device.

Parameters

inode	struttura dati sul file system che archivia e descrive attributi base su file, directory o qualsiasi altro oggetto
file_ptr	puntatore al descrittore file del device

Return values

0	se non si verifica nessun errore

9.25.2.8 GPIO_remove()

```
static int GPIO_remove (
            struct platform_device * pdev )  [static]
```

Viene chiamata automaticamente alla rimozione del modulo.

Parameters

pdev	

Return values

0	se non si verifica nessun errore

Dealloca tutta la memoria utilizzata dal driver, de-inizializzando il device e disattivando gli interrupt per il device, effettuando tutte le operazioni inverse della funzione GPIO_probe().

9.25.2.9 GPIO_write()

```
static ssize_t GPIO_write (
            struct file * file_ptr,
            const char * buf,
            size_t size,
            loff_t * off )  [static]
```

Invia dati al device.

Parameters

file_ptr	puntatore al descrittore file del device
buf	puntatore all'area di memoria dalla quale verranno copiati i count bytes
count	numeri di bytes da trasferire
off	long offset type che indica la posizione alla quale si sta effettuando l'accesso

9.25.2.10 module_platform_driver()

```
module_platform_driver (
              GPIO_driver  )
```

la macro module_platform_driver() prende in input la struttura platform_driver ed implementa le funzioni module↩
_init() e module_close() standard, chiamate quando il modulo viene caricato o rimosso dal kernel.

Parameters

GPIO_driver	struttura platform_driver associata al driver

9.25.3 Variable Documentation

9.25.3.1 __test_int_driver_id

```
const struct of_device_id __test_int_driver_id[]  [static]
```

Initial value:

```
={
    {.compatible = "GPIO"},
    {}
}
```

Identifica il device all'interno del device tree.

9.25.3.2 GPIO_driver

```
struct platform_driver GPIO_driver  [static]
```

Initial value:

```
= {
    .driver = {
                .name = DRIVER_NAME,
                .owner = THIS_MODULE,
                .of_match_table = of_match_ptr(__test_int_driver_id),
        },
    .probe = GPIO_probe,
    .remove = GPIO_remove
}
```

Definisce le funzioni probe() e remove() da chiamare al caricamento del driver.

9.25.3.3 GPIO_fops

```
struct file_operations GPIO_fops  [static]
```

Initial value:

```
= {
        .owner      = THIS_MODULE,
        .llseek     = GPIO_llseek,
        .read       = GPIO_read,
        .write      = GPIO_write,
        .poll       = GPIO_poll,
        .open       = GPIO_open,
        .release    = GPIO_release
}
```

Struttura che specifica le funzioni che agiscono sul device.

9.26 /media/saverio/OS/Users/Saverio/Desktop/SE/git/codici_da_mandare/FPGA/GPIO/↩ Driver/Kernel_Mode/GPIO_list.c File Reference

Permette di avere una serie di GPIO sotto lo stesso device.

9.26.1 Detailed Description

Permette di avere una serie di GPIO sotto lo stesso device.

9.26.2 Function Documentation

9.26.2.1 GPIO_list_add()

```
int GPIO_list_add (
            GPIO_list * list,
            GPIO * device )
```

Aggiunge un oggetto GPIO alla lista.

Parameters

list	puntatore a GPIO_list, lista a cui aggiungere l'oggetto
device	puntatore a GPIO, oggetto da aggiungere alla lista

Return values

-1	se è ststo già inserito il numero massimo di device
0	se non si manifesta nessun errore

9.26.2.2 GPIO_list_Destroy()

```
void GPIO_list_Destroy (
            GPIO_list* list )
```

Dealloca gli oggetti internamente contenuti nella GPIO_list.

Parameters

list	puntatore a GPIO_list, lista da distruggere

9.26.2.3 GPIO_list_device_count()

```
uint32_t GPIO_list_device_count (
            GPIO_list * list )
```

Restituisce il numero di device presenti nella lista.

Parameters

list	puntatore a GPIO_list, lista di cui si intende conoscere il numero di oggetti GPIO contenuti

Returns

numero di device presenti nella lista

9.26.2.4 GPIO_list_find_by_minor()

```
GPIO* GPIO_list_find_by_minor (
            GPIO_list * list,
            dev_t dev )
```

Ricerca un oggetto GPIO all'interno della lista tramite il minor number associato al device.

Parameters

list	puntatore a GPIO_list, lista in cui effettuare la ricerca
dev	major/minor number associato al device, parametro con cui viene invocata la open() o la release()

Returns

indirizzo dell'oggetto GPIO, se è presente nella lista, NULL altrimenti

9.26.2.5 GPIO_list_find_by_pdev()

```
GPIO* GPIO_list_find_by_pdev (
            GPIO_list * list,
            struct platform_device * pdev )
```

Ricerca un oggetto GPIO all'interno della lista tramite il campo pdev.

Parameters

list	puntatore a GPIO_list in cui effettuare la ricerca
pdev	puntatore a struct platform_device

Returns

indirizzo dell'oggetto GPIO, se è contenuto nella lista, NULL altrimenti

9.26.2.6 GPIO_list_find_irq_line()

```
GPIO* GPIO_list_find_irq_line (
            GPIO_list * list,
            int irq_line )
```

Ricerca un oggetto GPIO all'interno della lista tramite l' interrupt-number.

Parameters

list	puntatore a GPIO_list, lista in cui effettuare la ricerca
irq_line	linea di interruzione alla quale il device è connesso

Returns

indirizzo dell'oggetto GPIO, se è presente nella lista, NULL altrimenti

9.26.2.7 GPIO_list_Init()

```
int GPIO_list_Init (
            GPIO_list * list,
            uint32_t list_size )
```

Inizializza una struttura dati GPIO_list.

Parameters

list	puntatore a lista da inizializzare
list_size	numero massimo di device che la struttra dati potrà contenere

Return values

-ENOMEM	nel caso in cui la struttura non possa essere allocata in memoria
0	se non si manifestano errori

9.27 /media/saverio/OS/Users/Saverio/Desktop/SE/git/codici_da_mandare/FPGA/GPIO/↩ Driver/Kernel_Mode/GPIO_list.h File Reference

header file GPIO_list

Data Structures

- struct GPIO_list

  Struttura dati per la gestione di più device GPIO da parte del driver.

9.27.1 Detailed Description

header file GPIO_list

9.27.2 Function Documentation

9.27.2.1 GPIO_list_add()

```
int GPIO_list_add (
            GPIO_list * list,
            GPIO * device )
```

Aggiunge un oggetto GPIO alla lista.

Parameters

list	puntatore a GPIO_list, lista a cui aggiungere l'oggetto
device	puntatore a GPIO, oggetto da aggiungere alla lista

Return values

-1	se è stsso già inserito il numero massimo di device
0	se non si manifesta nessun errore

9.27.2.2 GPIO_list_Destroy()

```
void GPIO_list_Destroy (
            GPIO_list* list )
```

Dealloca gli oggetti internamente contenuti nella GPIO_list.

Parameters

list	puntatore a GPIO_list, lista da distruggere

9.27.2.3 GPIO_list_device_count()

```
uint32_t GPIO_list_device_count (
            GPIO_list * list )
```

Restituisce il numero di device presenti nella lista.

Parameters

list	puntatore a GPIO_list, lista di cui si intende conoscere il numero di oggetti GPIO contenuti

Returns

numero di device presenti nella lista

9.27.2.4 GPIO_list_find_by_minor()

```
GPIO* GPIO_list_find_by_minor (
            GPIO_list * list,
            dev_t dev )
```

Ricerca un oggetto GPIO all'interno della lista tramite il minor number associato al device.

Parameters

list	puntatore a GPIO_list, lista in cui effettuare la ricerca
dev	major/minor number associato al device, parametro con cui viene invocata la open() o la release()

Returns

indirizzo dell'oggetto GPIO, se è presente nella lista, NULL altrimenti

9.27.2.5 GPIO_list_find_by_pdev()

```
GPIO* GPIO_list_find_by_pdev (
            GPIO_list * list,
            struct platform_device * pdev )
```

Ricerca un oggetto GPIO all'interno della lista tramite il campo pdev.

Parameters

list	puntatore a GPIO_list in cui effettuare la ricerca
pdev	puntatore a struct platform_device

Returns

indirizzo dell'oggetto GPIO, se è contenuto nella lista, NULL altrimenti

9.27.2.6 GPIO_list_find_irq_line()

```
GPIO* GPIO_list_find_irq_line (
            GPIO_list * list,
            int irq_line )
```

Ricerca un oggetto GPIO all'interno della lista tramite l' interrupt-number.

Parameters

list	puntatore a GPIO_list, lista in cui effettuare la ricerca
irq_line	linea di interruzione alla quale il device è connesso

Returns

indirizzo dell'oggetto GPIO, se è presente nella lista, NULL altrimenti

9.27.2.7 GPIO_list_Init()

```
int GPIO_list_Init (
            GPIO_list * list,
            uint32_t list_size )
```

Inizializza una struttura dati GPIO_list.

Parameters

list	puntatore a lista da inizializzare
list_size	numero massimo di device che la struttra dati potrà contenere

Return values

-ENOMEM	nel caso in cui la struttura non possa essere allocata in memoria
0	se non si manifestano errori

9.28 /media/saverio/OS/Users/Saverio/Desktop/SE/git/codici_da_mandare/FPGA/GPIO/↩ Driver/UIO/GPIO_interrupt_uio_poll.c File Reference

permette la gestione del GPIO utilizzando un driver di tipo UIO

9.28.1 Detailed Description

permette la gestione del GPIO utilizzando un driver di tipo UIO

9.28.2 Function Documentation

9.28.2.1 read_reg()

```
unsigned int read_reg (
            void * addr,
            unsigned int offset )
```

Dealloca gli oggetti internamente contenuti nella GPIO_list.

Parameters

addr,puntatore	all' indirizzo da voler leggere
offset,offset	a partire dall' indirizzo a cui vogliamo scrivere

9.28.2.2 wait_for_interrupt()

```
void wait_for_interrupt (
            int fd0,
```

```
            int fd1,
            int fd2,
            void * addr_0,
            void * addr_1,
            void * addr_2 )
```

Dealloca gli oggetti internamente contenuti nella GPIO_list.

Parameters

fd0,valore	del file descriptor del primo GPIO
fd1,valore	del file descriptor del secondo GPIO
fd2,valore	del file descriptor del terzo GPIO
addr_0,indirizzo	base della prima periferica GPIO
addr_1,indirizzo	base della seconda periferica GPIO
addr_2,indirizzo	base della terza periferica GPIO

9.28.2.3 write_reg()

```
void write_reg (
            void * addr,
            unsigned int offset,
            unsigned int value )
```

Dealloca gli oggetti internamente contenuti nella GPIO_list.

Parameters

addr,puntatore	all' indirizzo da voler scrivere
offset,offset	a partire dall' indirizzo a cui vogliamo scrivere
value,valore	da voler scrivere

9.29 /media/saverio/OS/Users/Saverio/Desktop/SE/git/codici_da_mandare/FPGA/GPIO/↩ Driver/UIO/GPIO_interrupt_uio_poll.h File Reference

header file GPIO interrupt uio poll

9.29.1 Detailed Description

header file GPIO interrupt uio poll

9.29.2 Function Documentation

9.29.2.1 read_reg()

```
unsigned int read_reg (
            void * addr,
            unsigned int offset )
```

Dealloca gli oggetti internamente contenuti nella GPIO_list.

Parameters

addr,puntatore	all' indirizzo da voler leggere
offset,offset	a partire dall' indirizzo a cui vogliamo scrivere

Dealloca gli oggetti internamente contenuti nella GPIO_list.

Parameters

addr	indirizzo virtuale della periferica
offset	offset del registro a cui leggere

Returns

valore presente all'interno del registro

9.29.2.2 wait_for_interrupt()

```
void wait_for_interrupt (
            int fd0,
            int fd1,
            int fd2,
            void * addr_0,
            void * addr_1,
            void * addr_2 )
```

Dealloca gli oggetti internamente contenuti nella GPIO_list.

Parameters

fd0,valore	del file descriptor del primo GPIO
fd1,valore	del file descriptor del secondo GPIO
fd2,valore	del file descriptor del terzo GPIO
addr_0,indirizzo	base della prima periferica GPIO
addr_1,indirizzo	base della seconda periferica GPIO
addr_2,indirizzo	base della terza periferica GPIO

9.29.2.3 write_reg()

```
void write_reg (
             void * addr,
             unsigned int offset,
             unsigned int value )
```

Dealloca gli oggetti internamente contenuti nella GPIO_list.

Parameters

addr,puntatore	all' indirizzo da voler scrivere
offset,offset	a partire dall' indirizzo a cui vogliamo scrivere
value,valore	da voler scrivere

Dealloca gli oggetti internamente contenuti nella GPIO_list.

Parameters

addr	indirizzo virtuale della periferica
offset	offset del registro a cui scrivere
valore	da scrivere

9.30 /media/saverio/OS/Users/Saverio/Desktop/SE/git/codici_da_mandare/FPGA/GPIO/↩ Hardware/GPIO_1.0/hdl/GPIO_v1_0.vhd File Reference

Top level entity del custom IP core GPIO_V1_0_S00_AXI.VHD.

Entities

- GPIO_v1_0 entity
- arch_imp architecture

9.30.1 Detailed Description

Top level entity del custom IP core GPIO_V1_0_S00_AXI.VHD.

9.31 /media/saverio/OS/Users/Saverio/Desktop/SE/git/codici_da_mandare/FPGA/GPIO/↩ Hardware/GPIO_1.0/hdl/GPIO_v1_0_S00_AXI.vhd File Reference

Componente utilizzato collegare il GPIO al bus AXI e gestire le interruzioni.

Entities

- GPIO_v1_0_S00_AXI entity
- arch_imp architecture

9.31.1 Detailed Description

Componente utilizzato collegare il GPIO al bus AXI e gestire le interruzioni.

9.32 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Michele/FPGA/UART/Driver/Con↩ _interrupt/KERNEL_MODE/UART.c File Reference

Permette la comunicazione con la periferica UART.

9.32.1 Detailed Description

Permette la comunicazione con la periferica UART.

9.32.2 Function Documentation

9.32.2.1 UART_Destroy()

```
void UART_Destroy (
            UART* device )
```

Rimuove un device UART con le relative strutture kernel allocate per il suo funzionamento.

Parameters

device	puntatore a struttura UART che indica l'istanza UART da rimuovere

9.32.2.2 UART_GetData()

```
uint8_t UART_GetData (
            UART* device )
```

Restituisce il valore contenuto nel registro RX_REG del dispositivo UART specificato. dal parametro device.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare

Returns

valore contenuto nel registro ricezione del device

9.32.2.3 UART_GetDeviceAddress()

```
void* UART_GetDeviceAddress (
            UART* device )
```

Restituisce l'indirizzo virtuale di memoria cui è mappato un device.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare

9.32.2.4 UART_GetPollMask()

```
unsigned UART_GetPollMask (
            UART * device,
            struct file * file_ptr,
            struct poll_table_struct * wait )
```

Verifica che le operazioni di lettura risultino non-bloccanti.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare
file	puntatore al descrittore file del device
wait	puntatore alla struttura poll_table

Returns

maschera di bit che indica se sia possibile effettuare operazioni di lettura non bloccanti.

Back-end di tre diverse sys-calls: poll, epoll e select,

9.32.2.5 UART_GlobalInterruptDisable()

```
void UART_GlobalInterruptDisable (
            UART* device )
```

Disabilitazione interrupt globali.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare

9.32.2.6 UART_GlobalInterruptEnable()

```
void UART_GlobalInterruptEnable (
            UART* device )
```

Abilitazione interrupt globali.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare

9.32.2.7 UART_Init()

```
int UART_Init (
            UART* UART_device,
            struct module * owner,
            struct platform_device * pdev,
            struct class* class,
            const char* driver_name,
            const char* device_name,
            uint32_t serial,
            struct file_operations * f_ops,
            irq_handler_t irq_handler,
            uint32_t irq_mask )
```

Inizializza una struttura UART per il corrispondente device.

Parameters

UART_device	puntatore a struttura UART, corrispondente al device su cui operare
owner	puntatore a struttura struct module, proprietario del device (THIS_MODULE)
pdev	puntatore a struct platform_device
driver_name	nome del driver
device_name	nome del device
serial	numero seriale del device
f_ops	puntatore a struttura struct file_operations, specifica le funzioni che agiscono sul device
irq_handler	puntatore irq_handler_t alla funzione che gestisce gli interrupt generati dal device
irq_mask	maschera delle interruzioni attive del device

Return values

0	se non si è verificato nessun errore

9.32.2.8 UART_InterruptDisable()

```
void UART_InterruptDisable (
            UART* device,
            unsigned mask )
```

Disabilitazione interrupt per i singoli pin del device.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare
mask	maschera di selezione degli interrupt da disabilitare

9.32.2.9 UART_InterruptEnable()

```
void UART_InterruptEnable (
            UART* device,
            unsigned mask )
```

Abilitazione interrupt per i singoli pin del device.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare
mask	maschera di selezione degli interrupt da abilitare

9.32.2.10 UART_PendingInterrupt()

```
unsigned UART_PendingInterrupt (
            UART* device )
```

Fornisce una maschera che indica quali interrupt non sono ancora stati serviti e che quindi risultano pending.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare

Returns

maschera riportante gli interrupt che non sono stati ancora serviti

9.32.2.11 UART_ReadPollWakeUp()

```
void UART_ReadPollWakeUp (
            UART* device )
```

Risveglia i processi in attesa sulle code di read e poll.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare

9.32.2.12 UART_ResetCanRead()

```
void UART_ResetCanRead (
            UART* device )
```

Utilizzata per resettare il flag "can_read" di uno specifico device UART.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare

9.32.2.13 UART_ResetCanWrite()

```
void UART_ResetCanWrite (
            UART* device )
```

Utilizzata per resettare il flag "can_write" di uno specifico device UART.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare

9.32.2.14 UART_RXInterruptAck()

```
void UART_RXInterruptAck (
```

```
            UART* device )
```

Invia al device notifica di servizio dell'interrupt relativa alla ricezione.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare

9.32.2.15 UART_SetCanRead()

```
void UART_SetCanRead (
            UART* device )
```

Utilizzata per asserire il flag "can_read" di uno specifico device UART.

Parameters

device	puntatore a struttura UART, device su cui operare

9.32.2.16 UART_SetCanWrite()

```
void UART_SetCanWrite (
            UART* device )
```

Utilizzata per asserire il flag "can_write" di uno specifico device UART.

Parameters

device	puntatore a struttura UART, device su cui operare

9.32.2.17 UART_SetData()

```
void UART_SetData (
            UART* device,
            uint8_t dataToSend )
```

Inserisce all'interno del registro DATA_IN del dispositivo UART specificato tramite il parametro device il valore indicato nel parametro dataToSend.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare
dataToSend	valore da inserire all'interno del registro

9.32.2.18 UART_Start()

```
void UART_Start (
            UART* device )
```

Asserisce il segnale TX_EN iniziando la trasmissione.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare

9.32.2.19 UART_TestCanReadAndSleep()

```
void UART_TestCanReadAndSleep (
            UART* device )
```

Testa il valore del flag "can_read". Se è uguale a 0, ovvero non è possibile effettuare una lettura, mette in sleep il processo.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare

9.32.2.20 UART_TestCanWriteAndSleep()

```
void UART_TestCanWriteAndSleep (
            UART* device )
```

Testa il valore del flag "can_write". Se è uguale a 0, ovvero non è possibile effettuare una lettura, mette in sleep il processo.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare

9.32.2.21 UART_TXInterruptAck()

```
void UART_TXInterruptAck (
            UART* device )
```

Invia al device notifica di servizio dell'interrupt relativa alla trasmissione.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare

9.32.2.22 UART_WriteWakeUp()

```
void UART_WriteWakeUp (
            UART* device )
```

Risveglia i processi in attesa sulla coda di write.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare

9.33 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Michele/FPGA/UART/Driver/Con_interrupt/KERNEL_MODE/UART_kernel_main.c File Reference

Inizializza il driver kernel ed espone le funzionalità del modulo.

9.33.1 Detailed Description

Inizializza il driver kernel ed espone le funzionalità del modulo.

9.33.2 Function Documentation

9.33.2.1 module_platform_driver()

```
module_platform_driver (
            UART_driver  )
```

la macro module_platform_driver() prende in input la struttura platform_driver ed implementa le funzioni module_init() e module_close() standard, chiamate quando il modulo viene caricato o rimosso dal kernel.

Parameters

UART_driver	struttura platform_driver associata al driver

9.33.2.2 UART_irq_handler()

```
static irqreturn_t UART_irq_handler (
            int irq,
            struct pt_regs * regs )  [static]
```

Interrupt-handler.

Parameters

irq	Interrupt-number a cui il device è connesso
regs	registri sullo stack alla system call entry

Return values

IRQ_HANDLED	dopo aver servito l'interruzione

9.33.2.3 UART_llseek()

```
static loff_t UART_llseek (
            struct file * file_ptr,
            loff_t off,
            int whence )  [static]
```

Implementa le system-call lseek() e llseek().

Parameters

file_ptr	puntatore al descrittore file del device
off	offset da aggiungere al parametro whence per il posizionamento
whence	può assumere i valori SEEK_SET, SEEK_CUR o SEEK_END per specificare rispettivamente il riferimento dall'inizio file, dalla posizione corrente o dalla fine.

Returns

Nuova posizione della "testina" di lettura/scrittura

9.33.2.4 UART_open()

```
static int UART_open (
            struct inode * inode,
            struct file * file_ptr )  [static]
```

Invocata all'apertura del file corrispondente al device.

Return values

0	se non si verifica nessun errore

9.33.2.5 UART_poll()

```
static unsigned int UART_poll (
            struct file * file_ptr,
            struct poll_table_struct * wait )  [static]
```

Verifica che le operazioni di lettura risultino non-bloccanti.

Parameters

device	puntatore a struttura UART, che si riferisce al device su cui operare
file_ptr	puntatore al descrittore file del device
wait	puntatore alla struttura poll_table

Returns

maschera di bit che indica se sia possibile effettuare operazioni di lettura non bloccanti.

Back-end di tre diverse sys-calls: poll, epoll e select,

9.33.2.6 UART_read()

```
static ssize_t UART_read (
            struct file * file_ptr,
            char * buf,
            size_t count,
            loff_t * off )  [static]
```

Legge dati dal device.

Parameters

file_ptr	puntatore al descrittore file del device
buf	puntatore all'area di memoria dove verranno copiati i count bytes letti
count	numeri di bytes da trasferire
off	long offset type che indica la posizione alla quale si sta effettuando l'accesso

Note

l'aggiunta del flag O_NONBLOCK all'apertura del file descriptor associato al device farà sì che il processo chiamante non verrà bloccato se alla chiamata di una lettura non troverà dati disponibili

9.33.2.7 UART_release()

```
static int UART_release (
            struct inode * inode,
            struct file * file_ptr )  [static]
```

Invocata alla chiusura del file corrispondente al device.

Parameters

inode	struttura dati sul file system che archivia e descrive attributi base su file, directory o qualsiasi altro oggetto
file_ptr	puntatore al descrittore file del device

Return values

0	se non si verifica nessun errore

9.33.2.8 UART_remove()

```
static int UART_remove (
            struct platform_device * pdev )  [static]
```

Viene chiamata automaticamente alla rimozione del modulo.

Parameters

pdev	

Return values

0	se non si verifica nessun errore

Dealloca tutta la memoria utilizzata dal driver, de-inizializzando il device e disattivando gli interrupt per il device, effettuando tutte le operazioni inverse della funzione UART_probe().

9.33.2.9 UART_write()

```
static ssize_t UART_write (
            struct file * file_ptr,
            const char __user * buf,
            size_t count,
            loff_t * off )  [static]
```

Invia dati al device.

Parameters

file_ptr	puntatore al descrittore file del device
buf	puntatore all'area di memoria dalla quale verranno copiati i count bytes
count	numeri di bytes da trasferire
off	long offset type che indica la posizione alla quale si sta effettuando l'accesso

9.33.3 Variable Documentation

9.33.3.1 __test_int_driver_id

```
const struct of_device_id __test_int_driver_id[]  [static]
```

Initial value:

```
={
    {.compatible = "xlnx,UART-1.0"},
    {}
}
```

Identifica il device all'interno del device tree.

9.33.3.2 UART_driver

```
struct platform_driver UART_driver  [static]
```

Initial value:

```
= {
    .driver = {
                .name = DRIVER_NAME,
                .owner = THIS_MODULE,
                .of_match_table = of_match_ptr(__test_int_driver_id),
        },
    .probe = UART_probe,
    .remove = UART_remove
}
```

Definisce le funzioni probe() e remove() da chiamare al caricamento del driver.

9.33.3.3 UART_fops

```
struct file_operations UART_fops  [static]
```

Initial value:

```
= {
        .owner      = THIS_MODULE,
        .llseek     = UART_llseek,
        .read       = UART_read,
        .write      = UART_write,
        .poll       = UART_poll,
        .open       = UART_open,
        .release    = UART_release
}
```

Struttura che specifica le funzioni che agiscono sul device.

9.34 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Michele/FPGA/UART/Driver/Con↩ _interrupt/KERNEL_MODE/UART_list.c File Reference

Gestisce una lista di device UART.

9.34.1 Detailed Description

Gestisce una lista di device UART.

9.34.2 Function Documentation

9.34.2.1 UART_list_add()

```
int UART_list_add (
            UART_list * list,
            UART * device )
```

Aggiunge un oggetto UART alla lista.

Parameters

list	puntatore a UART_list, lista a cui aggiungere l'oggetto
device	puntatore a UART, oggetto da aggiungere alla lista

Return values

-1	se è ststo già inserito il numero massimo di device
0	se l'inserimento è avvenuto correttamente

9.34.2.2 UART_list_Destroy()

```
void UART_list_Destroy (
            UART_list* list )
```

Dealloca gli oggetti internamente contenuti nella UART_list.

Parameters

list	puntatore a UART_list, lista da distruggere

9.34.2.3 UART_list_device_count()

```
uint32_t UART_list_device_count (
            UART_list * list )
```

Restituisce il numero di device presenti nella lista.

Parameters

list	puntatore a UART_list, lista di cui si intende conoscere il numero di oggetti UART contenuti

Returns

numero di device presenti nella lista

9.34.2.4 UART_list_find_by_minor()

```
UART* UART_list_find_by_minor (
            UART_list * list,
            dev_t dev )
```

Ricerca un oggetto UART all'interno della lista tramite il minor number associato al device.

Parameters

list	puntatore a UART_list, lista in cui effettuare la ricerca
dev	major/minor number associato al device, parametro con cui viene invocata la open() o la release()

Returns

indirizzo dell'oggetto UART, se è presente nella lista, NULL altrimenti

9.34.2.5 UART_list_find_by_pdev()

```
UART* UART_list_find_by_pdev (
            UART_list * list,
            struct platform_device * pdev )
```

Ricerca un oggetto UART all'interno della lista tramite il campo pdev.

Parameters

list	puntatore a UART_list in cui effettuare la ricerca
pdev	puntatore a struct platform_device

Returns

indirizzo dell'oggetto UART, se è contenuto nella lista, NULL altrimenti

9.34.2.6 UART_list_find_irq_line()

```
UART* UART_list_find_irq_line (
            UART_list * list,
            int irq_line )
```

Ricerca un oggetto UART all'interno della lista tramite l' interrupt-number.

Parameters

list	puntatore a UART_list, lista in cui effettuare la ricerca
irq_line	linea di interruzione alla quale il device è connesso

Returns

indirizzo dell'oggetto UART, se è presente nella lista, NULL altrimenti

9.34.2.7 UART_list_Init()

```
int UART_list_Init (
            UART_list * list,
            uint32_t list_size )
```

Inizializza una struttura dati UART_list Istanzia una lista di dimensione pari a list_size dispositivi e inizializza i relativi puntatori al valore null.

Parameters

list	puntatore a lista da inizializzare
list_size	numero massimo di device che la struttra dati potrà contenere

Return values

-ENOMEM	nel caso in cui la struttura non possa essere allocata in memoria
0	se non si manifestano errori

9.35 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Michele/FPGA/UART/Driver/Con↩ _interrupt/KERNEL_MODE/UART_list.h File Reference

Header file UART_list.

Data Structures

- struct UART_list

  Struttura dati per la gestione di più device UART da parte del driver.

9.35.1 Detailed Description

Header file UART_list.

9.35.2 Function Documentation

9.35.2.1 UART_list_add()

```
int UART_list_add (
            UART_list * list,
            UART * device )
```

Aggiunge un oggetto UART alla lista.

Parameters

list	puntatore a UART_list, lista a cui aggiungere l'oggetto
device	puntatore a UART, oggetto da aggiungere alla lista

Return values

-1	se è ststo già inserito il numero massimo di device
0	se l'inserimento è avvenuto correttamente

9.35.2.2 UART_list_Destroy()

```
void UART_list_Destroy (
            UART_list* list )
```

Dealloca gli oggetti internamente contenuti nella UART_list.

Parameters

list	puntatore a UART_list, lista da distruggere

9.35.2.3 UART_list_device_count()

```
uint32_t UART_list_device_count (
            UART_list * list )
```

Restituisce il numero di device presenti nella lista.

Parameters

list	puntatore a UART_list, lista di cui si intende conoscere il numero di oggetti UART contenuti

Returns

numero di device presenti nella lista

9.35.2.4 UART_list_find_by_minor()

```
UART* UART_list_find_by_minor (
            UART_list * list,
            dev_t dev )
```

Ricerca un oggetto UART all'interno della lista tramite il minor number associato al device.

Parameters

list	puntatore a UART_list, lista in cui effettuare la ricerca
dev	major/minor number associato al device, parametro con cui viene invocata la open() o la release()

Returns

indirizzo dell'oggetto UART, se è presente nella lista, NULL altrimenti

9.35.2.5 UART_list_find_by_pdev()

```
UART* UART_list_find_by_pdev (
            UART_list * list,
            struct platform_device * pdev )
```

Ricerca un oggetto UART all'interno della lista tramite il campo pdev.

Parameters

list	puntatore a UART_list in cui effettuare la ricerca
pdev	puntatore a struct platform_device

Returns

indirizzo dell'oggetto UART, se è contenuto nella lista, NULL altrimenti

9.35.2.6 UART_list_find_irq_line()

```
UART* UART_list_find_irq_line (
            UART_list * list,
            int irq_line )
```

Ricerca un oggetto UART all'interno della lista tramite l' interrupt-number.

Parameters

list	puntatore a UART_list, lista in cui effettuare la ricerca
irq_line	linea di interruzione alla quale il device è connesso

Returns

indirizzo dell'oggetto UART, se è presente nella lista, NULL altrimenti

9.35.2.7 UART_list_Init()

```
int UART_list_Init (
            UART_list * list,
            uint32_t list_size )
```

Inizializza una struttura dati UART_list Istanzia una lista di dimensione pari a list_size dispositivi e inizializza i relativi puntatori al valore null.

Parameters

list	puntatore a lista da inizializzare
list_size	numero massimo di device che la struttra dati potrà contenere

Return values

-ENOMEM	nel caso in cui la struttura non possa essere allocata in memoria
0	se non si manifestano errori

9.36 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Michele/FPGA/UART/Driver/Con↩ _interrupt/UIO/UART_interrupt_uio.c File Reference

permette la gestione della periferica UART utilizzando un driver di tipo UIO

9.36.1 Detailed Description

permette la gestione della periferica UART utilizzando un driver di tipo UIO

9.36.2 Function Documentation

9.36.2.1 read_reg()

```
unsigned int read_reg (
            void * addr,
            unsigned int offset )
```

Utilizzata per leggere un valore da un registro della periferica, specificando l'indirizzo base virtuale e l'offset del registro da cui leggere.

Dealloca gli oggetti internamente contenuti nella GPIO_list.

Parameters

addr	indirizzo virtuale della periferica
offset	offset del registro a cui leggere

Returns

valore presente all'interno del registro

9.36.2.2 wait_for_interrupt()

```
void wait_for_interrupt (
            int file_descr,
            void * addr )
```

Attende l' arrivo di un interrupt utilizzando la read su un device UIO.

Parameters

file_descr	descrittore del UIO driver
addr	indirizzo virtuale della periferica

9.36.2.3 write_reg()

```
void write_reg (
            void * addr,
            unsigned int offset,
            unsigned int value )
```

Utilizzata per scrivere un valore all'interno di un registro della periferica, specificando l'indirizzo base virtuale e l'offset del registro in cui scrivere.

Dealloca gli oggetti internamente contenuti nella GPIO_list.

Parameters

addr	indirizzo virtuale della periferica
offset	offset del registro a cui scrivere
valore	da scrivere

9.37 /media/saverio/OS/Users/Saverio/Desktop/SE/git/Michele/FPGA/UART/Driver/Con_interrupt/UIO/UART_interrupt_uio.h File Reference

header file UART_interrupt_uio

9.37.1 Detailed Description

header file UART_interrupt_uio

9.37.2 Function Documentation

9.37.2.1 read_reg()

```
unsigned int read_reg (
            void * addr,
            unsigned int offset )
```

Dealloca gli oggetti internamente contenuti nella GPIO_list.

Parameters

addr,puntatore	all' indirizzo da voler leggere
offset,offset	a partire dall' indirizzo a cui vogliamo scrivere

Dealloca gli oggetti internamente contenuti nella GPIO_list.

Parameters

addr	indirizzo virtuale della periferica
offset	offset del registro a cui leggere

Returns

valore presente all'interno del registro

9.37.2.2 wait_for_interrupt()

```
void wait_for_interrupt (
            int file_descr,
            void * addr )
```

Attende l' arrivo di un interrupt utilizzando la read su un device UIO.

Parameters

file_descr	descrittore del UIO driver
addr	indirizzo virtuale della periferica

9.37.2.3 write_reg()

```
void write_reg (
            void * addr,
            unsigned int offset,
            unsigned int value )
```

Dealloca gli oggetti internamente contenuti nella GPIO_list.

Parameters

addr,puntatore	all' indirizzo da voler scrivere
offset,offset	a partire dall' indirizzo a cui vogliamo scrivere
value,valore	da voler scrivere

Dealloca gli oggetti internamente contenuti nella GPIO_list.

Parameters

addr	indirizzo virtuale della periferica
offset	offset del registro a cui scrivere
valore	da scrivere

Index