Crosmi

1.0

Generated by Doxygen 1.8.11

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 Buffer Struct Reference

Data Fields

- uint32_t size
- uint8_t buffer [BUFFER_SIZE]

3.1.1 Field Documentation

3.1.1.1 uint8_t buffer[BUFFER_SIZE]

3.1.1.2 uint32_t size

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

- ourUsart.c

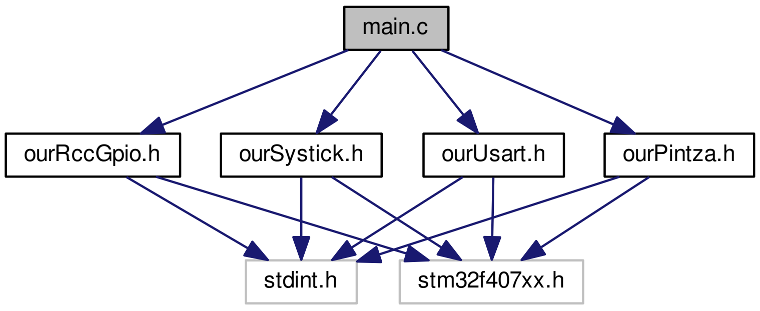
Chapter 4

File Documentation

4.1 main.c File Reference

```
#include "ourUsart.h"
#include "ourPintza.h"
#include "ourRccGpio.h"
#include "ourSystick.h"
```

Include dependency graph for main.c:



Functions

- void irakurri (void)
- void initGPIO (void)
- int main ()

Variables

- uint32_t piztuta = 1

4.1.1 Function Documentation

4.1.1.1 void initGPIO (void)

4.1.1.2 void irakurri (void)

4.1.1.3 int main ( )

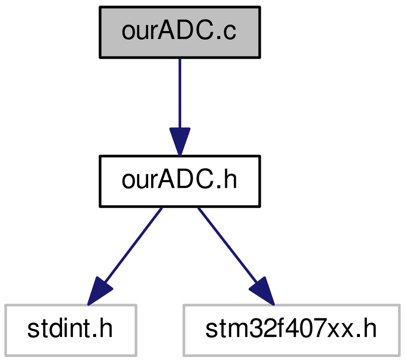
4.1.2 Variable Documentation

4.1.2.1 uint32_t piztuta = 1

4.2 ourADC.c File Reference

```
#include "ourADC.h"
```

Include dependency graph for ourADC.c:



Functions

- void initADC (uint32_t timer2TRGO, uint32_t interrupzioa, uint32_t kanala)
- void switchADC (int piztu)
- void setADCCallBack (void(∗funtzioa)(uint16_t))
- void ourADCHandler ()
- uint16_t getAzkenBalioa ()
- uint16_t getBalioa (void)

Variables

- uint16_t azkenBalioa = 0
- void(∗ callback)(uint16_t)=0

4.2.1 Function Documentation

4.2.1.1 uint16_t getAzkenBalioa (void)

Function that returns the last value of ADC.

Returns

azkenBalioa: uint16_t type ADC last value.

4.2.1.2 uint16_t getBalioa (void)

This function return the last value.

Returns

azkenBalioa: uint16_t type conversion's last value.

4.2.1.3 void initADC (uint32_t *timer2TRGO,* uint32_t *interrupzioa,* uint32_t *kanala*)

This function initializes ADC, setting core basic params to launch ADC. Some of the params are optional configurations such as interruption and TRGO.

Parameters

timer2TRGO	uint32_t type but acts as a boolean to know if timer 2's TRGO is needed.
interrupzioa	uint32_t type but acts as boolean to know if interruption is needed.
kanala	uint32_t type holds the value of which channel should be used by the ADC.

Returns

void.

4.2.1.4 void ourADCHandler (void)

This function acts as a handler for the interruption caused by the ADC.

Returns

void.

4.2.1.5 void setADCCallBack (void(∗)(uint16_t) *funtzioa*)

This function sets a pointer to another function which accepts a paramater of type uint16_t. Callback funtcion. funtzio: void type points to a function.

Returns

void

4.2.1.6 void switchADC (int *piztu*)

This function enable or disable ADC peripheral depending on the given parameter that acts as a boolean type.

Parameters

piztu	int type but acts as a boolean to know whether is needed to enable or disable ADC1.

Returns

void.

4.2.2 Variable Documentation

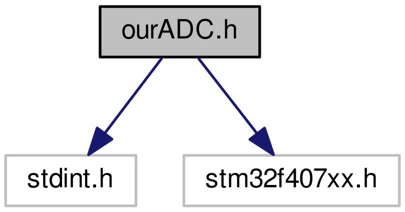
4.2.2.1 uint16_t azkenBalioa = 0

4.2.2.2 void(∗ callback) (uint16_t)=0

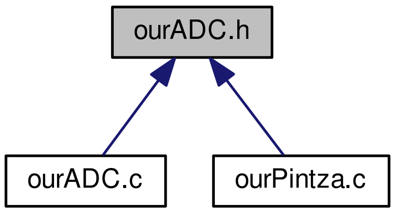
4.3 ourADC.h File Reference

```
#include <stdint.h>
#include <stm32f407xx.h>
```

Include dependency graph for ourADC.h:



This graph shows which files directly or indirectly include this file:



Functions

- void initADC (uint32_t timer2TRGO, uint32_t interrupzioa, uint32_t kanala)
- void switchADC (int piztu)
- uint16_t getAzkenBalioa (void)
- uint16_t getBalioa (void)
- void ourADCHandler (void)
- void setADCCallBack (void(∗funtzioa)(uint16_t))

4.3.1 Function Documentation

4.3.1.1 uint16_t getAzkenBalioa (void)

Function that returns the last value of ADC.

Returns

azkenBalioa: uint16_t type ADC last value.

4.3.1.2 uint16_t getBalioa (void)

This function return the last value.

Returns

azkenBalioa: uint16_t type conversion's last value.

4.3.1.3 void initADC (uint32_t *timer2TRGO,* uint32_t *interrupzioa,* uint32_t *kanala*)

This function initializes ADC, setting core basic params to launch ADC. Some of the params are optional configurations such as interruption and TRGO.

Parameters

timer2TRGO	uint32_t type but acts as a boolean to know if timer 2's TRGO is needed.
interrupzioa	uint32_t type but acts as boolean to know if interruption is needed.
kanala	uint32_t type holds the value of which channel should be used by the ADC.

Returns

void.

4.3.1.4 void ourADCHandler (void)

This function acts as a handler for the interruption caused by the ADC.

Returns

void.

4.3.1.5 void setADCCallBack (void(∗)(uint16_t) *funtzioa*)

This function sets a pointer to another function which accepts a paramater of type uint16_t. Callback funtcion. funtzio: void type points to a function.

Returns

void

4.3.1.6 void switchADC (int *piztu*)

This function enable or disable ADC peripheral depending on the given parameter that acts as a boolean type.

Parameters

piztu	int type but acts as a boolean to know whether is needed to enable or disable ADC1.

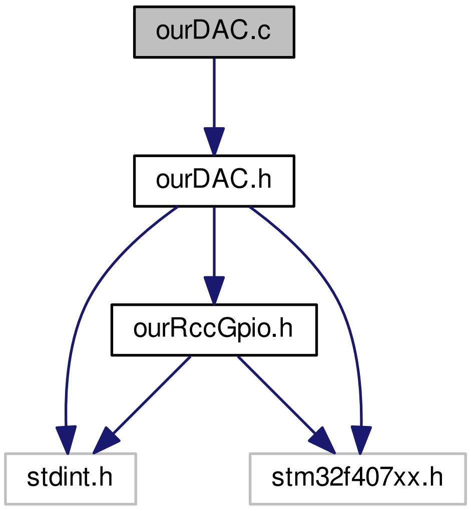
Returns

void.

4.4 ourDAC.c File Reference

```
#include "ourDAC.h"
```

Include dependency graph for ourDAC.c:



Functions

- void initGPIODAC2 (void)
- void initDAC (uint32_t dma, uint32_t trgo2)
- void setBalioa (uint32_t balioa)

4.4.1 Function Documentation

4.4.1.1 void initDAC (uint32_t *dma,* uint32_t *trgo2*)

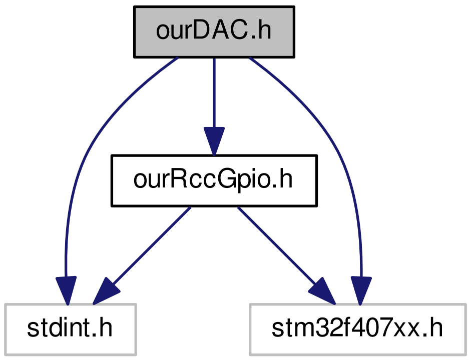
4.4.1.2 void initGPIODAC2 (void)

4.4.1.3 void setBalioa (uint32_t *balioa*)

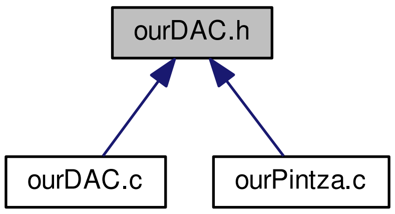
4.5 ourDAC.h File Reference

```
#include <stdint.h>
#include "ourRccGpio.h"
#include <stm32f407xx.h>
```

Include dependency graph for ourDAC.h:



This graph shows which files directly or indirectly include this file:



Functions

- void initDAC (uint32_t dma, uint32_t trgo2)
- void setBalioa (uint32_t balioa)

4.5.1 Function Documentation

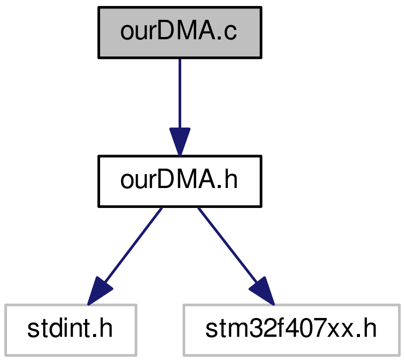
4.5.1.1 void initDAC (uint32_t *dma,* uint32_t *trgo2*)

4.5.1.2 void setBalioa (uint32_t *balioa*)

4.6 ourDMA.c File Reference

```
#include "ourDMA.h"
```

Include dependency graph for ourDMA.c:



Functions

- void initDMA2DAC (uint16_t ∗emaitza)

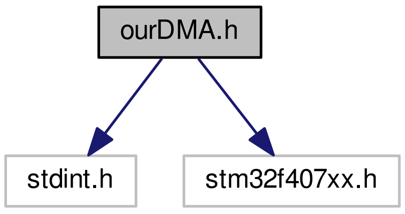
4.6.1 Function Documentation

4.6.1.1 void initDMA2DAC (uint16_t ∗ *emaitza*)

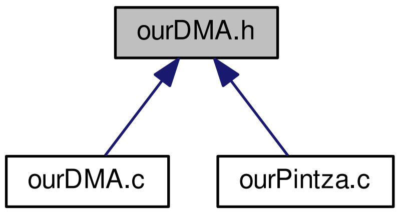
4.7 ourDMA.h File Reference

```
#include <stdint.h>
#include <stm32f407xx.h>
```

Include dependency graph for ourDMA.h:



This graph shows which files directly or indirectly include this file:



Functions

- void initDMA2DAC (uint16_t ∗emaitza)

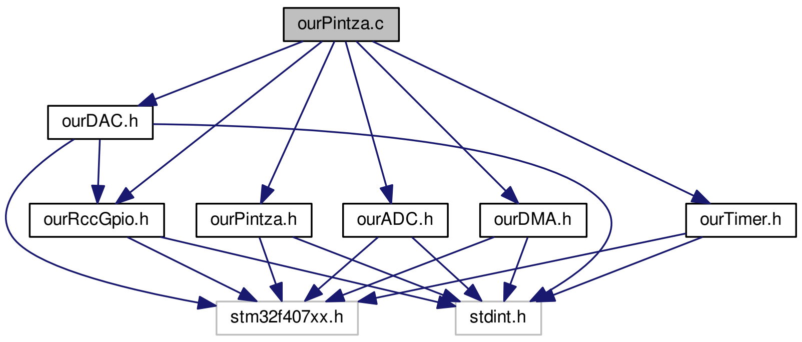
4.7.1 Function Documentation

4.7.1.1 void initDMA2DAC (uint16_t ∗ *emaitza*)

4.8 ourPintza.c File Reference

```
#include "ourPintza.h"
#include "ourADC.h"
#include "ourTimer.h"
#include "ourRccGpio.h"
#include "ourDMA.h"
#include "ourDAC.h"
```

Include dependency graph for ourPintza.c:



Macros

- #define ZIKLO_KOP 100
- #define TIMER_ABIADURA 10
- #define OFFSET (uint16_t) 0x04d9

Functions

- void ADCcallback (uint16_t balioa)
- void initGPIOA6 (void)
- void initPintza (void)
- void powerPintza (uint32_t piztu)
- uint16_t getAzkenKontsumoa ()
- void setPintzaCallback (void(∗funtzioa)(uint16_t))

Variables

- uint16_t maxBalioa = 0
- uint32_t zikloak = 0
- uint16_t azkenMaxBalioa = 0
- void(∗ callbackPintza)(uint16_t)=0

4.8.1 Macro Definition Documentation

4.8.1.1 #define OFFSET (uint16_t) 0x04d9

4.8.1.2 #define TIMER_ABIADURA 10

4.8.1.3 #define ZIKLO_KOP 100

4.8.2 Function Documentation

4.8.2.1 void ADCcallback (uint16_t *balioa*)

This function calculates the max value converted by the ADC. It waits for 3000 cycles and takes the max value. Used to take the peak value of the alternate current. And sets the max value to the callback function.

Parameters

balio	uint16_t type current value converted by the ADC.

Returns

void.

4.8.2.2 uint16_t getAzkenKontsumoa (void)

Gets the last value from the conversion made by ADC.

Returns

azkenBalio: uint16_t type last conversion value.

4.8.2.3 void initGPIOA6 (void)

Initializes GPIO6 and sets its mode to Analog mode.

Returns

void.

4.8.2.4 void initPintza (void)

This function initializes some peripherals to work with an extern tool (pintza). Initializes GPIO6, Timer 2, ADC, DAC and DMA.

Returns

void.

4.8.2.5 void powerPintza (uint32_t *piztu*)

This function enables or disables Timer2 and ADC depending on the given param.

Parameters

piztu	uint32_t type acts as a boolean to enable or disable Timer 2 and ADC.

Returns

void.

4.8.2.6 void setPintzaCallback (void(∗)(uint16_t) *funtzioa*)

Sets callback function that recieves the last value of the conversion as parameter.

Parameters

funtzioa	void type function, is a pointer to a void returning function which take uint16_t type parameter.

Returns

void.

4.8.3 Variable Documentation

4.8.3.1 uint16_t azkenMaxBalioa = 0

4.8.3.2 void(∗ callbackPintza) (uint16_t)=0

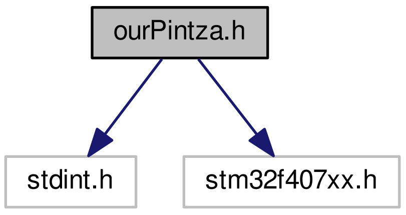
4.8.3.3 uint16_t maxBalioa = 0

4.8.3.4 uint32_t zikloak = 0

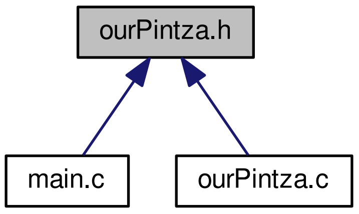
4.9 ourPintza.h File Reference

```
#include <stdint.h>
#include <stm32f407xx.h>
```

Include dependency graph for ourPintza.h:



This graph shows which files directly or indirectly include this file:



Functions

- void initPintza (void)
- void powerPintza (uint32_t piztu)
- uint16_t getAzkenKontsumoa (void)
- void setPintzaCallback (void(∗funtzioa)(uint16_t))
- void ADCcallback (uint16_t balioa)

4.9.1 Function Documentation

4.9.1.1 void ADCcallback (uint16_t *balioa*)

This function calculates the max value converted by the ADC. It waits for 3000 cycles and takes the max value. Used to take the peak value of the alternate current. And sets the max value to the callback function.

Parameters

balio	uint16_t type current value converted by the ADC.

Returns

void.

4.9.1.2 uint16_t getAzkenKontsumoa (void)

Gets the last value from the conversion made by ADC.

Returns

azkenBalio: uint16_t type last conversion value.

4.9.1.3 void initPintza (void)

This function initializes some peripherals to work with an extern tool (pintza). Initializes GPIO6, Timer 2, ADC, DAC and DMA.

Returns

void.

4.9.1.4 void powerPintza (uint32_t *piztu*)

This function enables or disables Timer2 and ADC depending on the given param.

Parameters

piztu	uint32_t type acts as a boolean to enable or disable Timer 2 and ADC.

Returns

void.

4.9.1.5 void setPintzaCallback (void(∗)(uint16_t) *funtzioa*)

Sets callback function that recieves the last value of the conversion as parameter.

Parameters

funtzioa	void type function, is a pointer to a void returning function which take uint16_t type parameter.

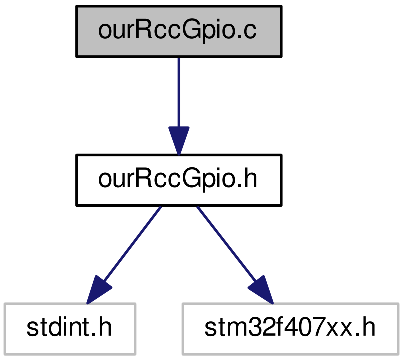
Returns

void.

4.10 ourRccGpio.c File Reference

`#include "ourRccGpio.h"`

Include dependency graph for ourRccGpio.c:



Functions

- void RCC_AHB1PeriphClockCmd (uint32_t nPeriph, uint32_t on)
- void RCC_AHB1APB2PeriphClockCmd (uint32_t nPeriph, uint32_t on)

- void initGpioPinMode (GPIO_TypeDef ∗gpio, uint32_t pin, GPIOMode_Type mode)
- void togleGpioPinValue (GPIO_TypeDef ∗gpio, uint32_t pin)
- void setGpioPinValue (GPIO_TypeDef ∗gpio, uint32_t pin, uint32_t value)
- uint32_t getGpioPinValue (GPIO_TypeDef ∗gpio, uint32_t pin)
- void setGpioPinAF (GPIO_TypeDef ∗gpio, uint32_t pin, uint32_t AF)

4.10.1 Function Documentation

4.10.1.1 uint32_t getGpioPinValue (GPIO_TypeDef ∗ *gpio,* uint32_t *pin*)

4.10.1.2 void initGpioPinMode (GPIO_TypeDef ∗ *gpio,* uint32_t *pin,* GPIOMode_Type *mode*)

4.10.1.3 void RCC_AHB1APB2PeriphClockCmd (uint32_t *nPeriph,* uint32_t *on*)

4.10.1.4 void RCC_AHB1PeriphClockCmd (uint32_t *nPeriph,* uint32_t *on*)

4.10.1.5 void setGpioPinAF (GPIO_TypeDef ∗ *gpio,* uint32_t *pin,* uint32_t *AF*)

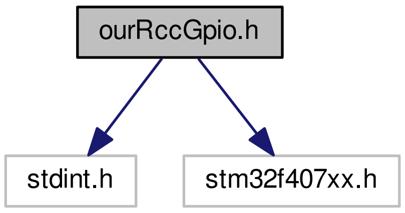
4.10.1.6 void setGpioPinValue (GPIO_TypeDef ∗ *gpio,* uint32_t *pin,* uint32_t *value*)

4.10.1.7 void togleGpioPinValue (GPIO_TypeDef ∗ *gpio,* uint32_t *pin*)

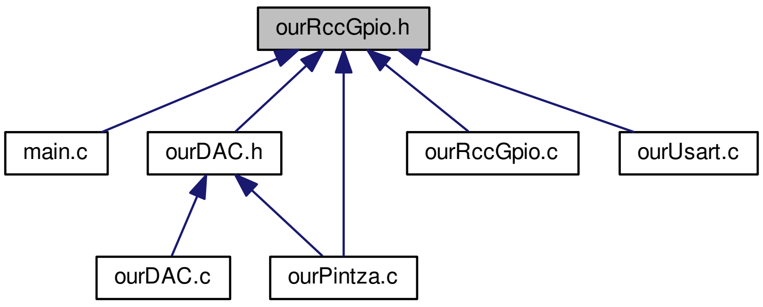
4.11 ourRccGpio.h File Reference

```
#include <stdint.h>
#include <stm32f407xx.h>
```

Include dependency graph for ourRccGpio.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define RCC_AHB1Periph_GPIOA ((uint32_t)0x01)
- #define RCC_AHB1Periph_GPIOC ((uint32_t)(0x01<<2))
- #define RCC_AHB1Periph_GPIOF ((uint32_t)(0x01<<5))
- #define RCC_AHB1Periph_GPIOD ((uint32_t)(0x01<<3))
- #define RCC_AHB1APB2Periph_SYSCFG ((uint32_t)(0x01<<14))

Enumerations

- enum GPIOMode_Type { GPIO_Mode_IN = 0x00, GPIO_Mode_OUT = 0x01, GPIO_Mode_AF = 0x02, G↩
  PIO_Mode_AN = 0x03 }

Functions

- void RCC_AHB1PeriphClockCmd (uint32_t nPeriph, uint32_t on)
- void RCC_AHB1APB2PeriphClockCmd (uint32_t nPeriph, uint32_t on)
- void initGpioPinMode (GPIO_TypeDef ∗, uint32_t pin, GPIOMode_Type mode)
- void togleGpioPinValue (GPIO_TypeDef ∗, uint32_t pin)
- void setGpioPinValue (GPIO_TypeDef ∗, uint32_t pin, uint32_t value)
- uint32_t getGpioPinValue (GPIO_TypeDef ∗, uint32_t pin)
- void setGpioPinAF (GPIO_TypeDef ∗gpio, uint32_t pin, uint32_t AF)

4.11.1 Macro Definition Documentation

4.11.1.1 #define RCC_AHB1APB2Periph_SYSCFG ((uint32_t)(0x01<<14))

4.11.1.2 #define RCC_AHB1Periph_GPIOA ((uint32_t)0x01)

4.11.1.3 #define RCC_AHB1Periph_GPIOC ((uint32_t)(0x01<<2))

4.11.1.4 #define RCC_AHB1Periph_GPIOD ((uint32_t)(0x01<<3))

4.11.1.5 #define RCC_AHB1Periph_GPIOF ((uint32_t)(0x01<<5))

4.11.2 Enumeration Type Documentation

4.11.2.1 enum GPIOMode_Type

Enumerator

GPIO_Mode_IN	GPIO Input Mode
GPIO_Mode_OUT	GPIO Output Mode
GPIO_Mode_AF	GPIO Alternate function Mode
GPIO_Mode_AN	GPIO Analog Mode

4.11.3 Function Documentation

4.11.3.1 uint32_t getGpioPinValue (GPIO_TypeDef $*$, uint32_t *pin*)

4.11.3.2 void initGpioPinMode (GPIO_TypeDef $*$, uint32_t *pin,* GPIOMode_Type *mode*)

4.11.3.3 void RCC_AHB1APB2PeriphClockCmd (uint32_t *nPeriph,* uint32_t *on*)

4.11.3.4 void RCC_AHB1PeriphClockCmd (uint32_t *nPeriph,* uint32_t *on*)

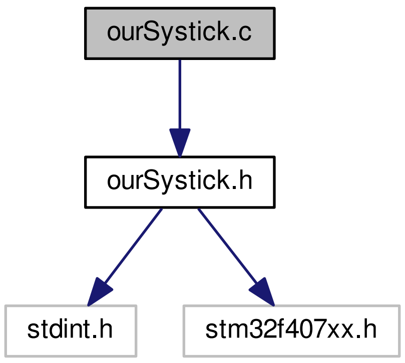
4.11.3.5 void setGpioPinAF (GPIO_TypeDef $*$ *gpio,* uint32_t *pin,* uint32_t *AF*)

4.11.3.6 void setGpioPinValue (GPIO_TypeDef $*$, uint32_t *pin,* uint32_t *value*)

4.11.3.7 void togleGpioPinValue (GPIO_TypeDef $*$, uint32_t *pin*)

4.12 ourSystick.c File Reference

```
#include "ourSystick.h"
```
Include dependency graph for ourSystick.c:



Functions

- void initSysTick (uint32_t ms, uint32_t internalClk)
- uint32_t getSysTicks (void)
- void waitNextSysTick (void)
- void ourSysTickHandler (void)

Variables

- uint32_t systicks = 0
- uint32_t systickOld = 0

4.12.1 Function Documentation

4.12.1.1 uint32_t getSysTicks (void)

Get the current system tick value. systicks: uint32_t type ticks value.

4.12.1.2 void initSysTick (uint32_t *ms*, uint32_t *internalClk*)

Initializes the System Tick core peripheral.

Parameters

ms	uint32_t type sets the milliseconds for the clock
internalClk	uint32_t type optional parameter, acts as a boolean to know whether internal clock is needed.

Returns

void.

4.12.1.3 void ourSysTickHandler (void)

Interruption handler for SysTick.

Returns

void.

4.12.1.4 void waitNextSysTick (void)

This function is used to count until the next system tick.

Returns

void.

4.12.2 Variable Documentation

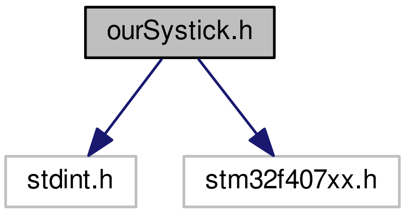
4.12.2.1 uint32_t systickOld = 0

4.12.2.2 uint32_t systicks = 0

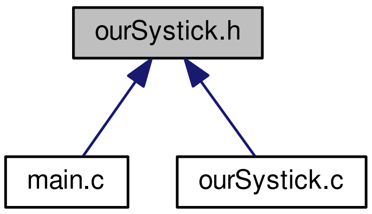
4.13 ourSystick.h File Reference

```
#include <stdint.h>
#include <stm32f407xx.h>
```

Include dependency graph for ourSystick.h:



This graph shows which files directly or indirectly include this file:



Functions

- void initSysTick (uint32_t ms, uint32_t internalClk)
- uint32_t getSysTicks (void)
- void waitNextSysTick (void)
- void ourSysTickHandler (void)

4.13.1 Function Documentation

4.13.1.1 uint32_t getSysTicks (void)

Get the current system tick value. systicks: uint32_t type ticks value.

4.13.1.2 void initSysTick (uint32_t *ms*, uint32_t *internalClk*)

Initializes the System Tick core peripheral.

Parameters

ms	uint32_t type sets the milliseconds for the clock
internalClk	uint32_t type optional parameter, acts as a boolean to know whether internal clock is needed.

Returns

void.

4.13.1.3 void ourSysTickHandler (void)

Interruption handler for SysTick.

Returns

void.

4.13.1.4 void waitNextSysTick (void)

This function is used to count until the next system tick.

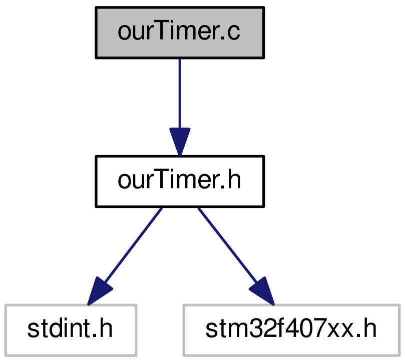
Returns

void.

4.14 ourTimer.c File Reference

```
#include "ourTimer.h"
```

Include dependency graph for ourTimer.c:



Functions

- void initTimer2 (uint32_t ms, uint32_t trGo, uint32_t interrupzioa)
- void switchTimer2 (int piztu)
- void waitTick (void)
- uint32_t getTicks (void)
- void setTimer2CallBack (void(∗funtzioa)(void))
- void ourTimer2Handler (void)

Variables

- void(∗ callbackTimer2)(void)=0
- volatile uint32_t ticks =0
- volatile uint32_t ticksOld =0

4.14.1 Function Documentation

4.14.1.1 uint32_t getTicks (void)

4.14.1.2 void initTimer2 (uint32_t *ms*, uint32_t *trGo*, uint32_t *interrupzioa*)

Initializes timer 2 setting basic configurations for launch. Some given paramaters such as trGo and interrupzio are optional.

Parameters

ms	uint32_t type time in millisecond above which timer 2 have to operate.
trGo	uint32_t acts as boolean to know whether is needed to enable TRGO signal.
interrupzioa	uint32_t acts as a boolean to know whether interruption configuration is needed.

Returns

void.

4.14.1.3 void ourTimer2Handler (void)

Custom handler for Timer 2 interruption.

Returns

void.

4.14.1.4 void setTimer2CallBack (void(∗)(void) *funtzioa*)

4.14.1.5 void switchTimer2 (int *piztu*)

Enables or disables the Timer 2 depending on the given parameter.

Parameters

piztu	uint32_t type acts as a boolean to enable or disable the Timer 2.

Returns

void.

4.14.1.6 void waitTick (void)

4.14.2 Variable Documentation

4.14.2.1 void(∗ callbackTimer2) (void)=0

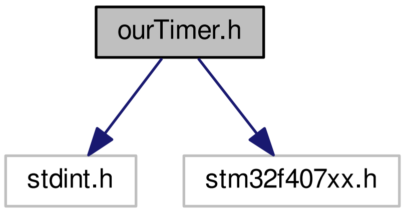
4.14.2.2 volatile uint32_t ticks =0

4.14.2.3 volatile uint32_t ticksOld =0

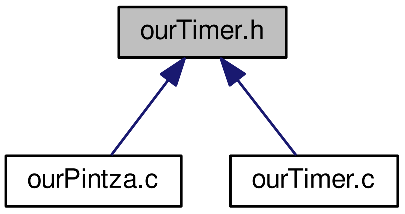
4.15 ourTimer.h File Reference

```
#include <stdint.h>
#include <stm32f407xx.h>
```

Include dependency graph for ourTimer.h:



This graph shows which files directly or indirectly include this file:



Functions

- void initTimer2 (uint32_t ms, uint32_t trGo, uint32_t interrupzioa)
- void switchTimer2 (int piztu)
- void waitTick (void)
- uint32_t getTicks (void)
- void ourTimer2Handler (void)
- void setTimer2CallBack (void(∗funtzioa)(void))

4.15.1 Function Documentation

4.15.1.1 uint32_t getTicks (void)

4.15.1.2 void initTimer2 (uint32_t *ms,* uint32_t *trGo,* uint32_t *interrupzioa*)

Initializes timer 2 setting basic configurations for launch. Some given paramaters such as trGo and interrupzio are optional.

Parameters

ms	uint32_t type time in millisecond above which timer 2 have to operate.
trGo	uint32_t acts as boolean to know whether is needed to enable TRGO signal.
interrupzioa	uint32_t acts as a boolean to know whether interruption configuration is needed.

Returns

void.

4.15.1.3 void ourTimer2Handler (void)

Custom handler for Timer 2 interruption.

Returns

void.

4.15.1.4 void setTimer2CallBack (void(∗)(void) *funtzioa*)

4.15.1.5 void switchTimer2 (int *piztu*)

Enables or disables the Timer 2 depending on the given parameter.

Parameters

piztu	uint32_t type acts as a boolean to enable or disable the Timer 2.

Returns

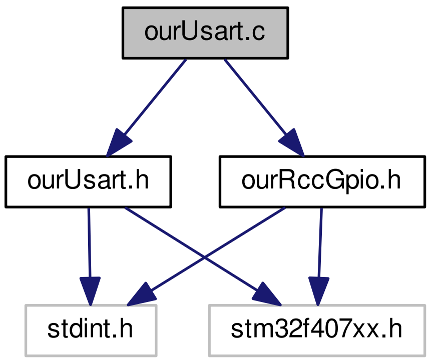
void.

4.15.1.6 void waitTick (void)

4.16 ourUsart.c File Reference

```
#include "ourUsart.h"
#include "ourRccGpio.h"
```

Include dependency graph for ourUsart.c:



Data Structures

- struct Buffer

Macros

- #define USART3_TX 8
- #define USART3_RX 9
- #define BUFFER_SIZE 20

Typedefs

- typedef struct Buffer Buffer

Functions

- void initGPIOUsart3 (void)
- void pushBuffer (Buffer ∗buffer, uint8_t balioa)
- uint8_t popBuffer (Buffer ∗buffer)
- uint32_t readBufferSize ()
- void initUsart3 (uint32_t baudRate, uint32_t interrupzioak)
- void writeUart3Blocking (uint8_t ∗mezua, uint32_t luzera)
- void writeByte (uint8_t mezua)
- void writeUart3 (uint8_t ∗mezua, uint32_t luzera)
- uint32_t readUart3 (uint8_t ∗pMsg, uint32_t maxLen)
- void ourUSART3Handler ()

Variables

- Buffer bufferIdatzi
- Buffer bufferIrakurri

4.16.1 Macro Definition Documentation

4.16.1.1 #define BUFFER_SIZE 20

4.16.1.2 #define USART3_RX 9

4.16.1.3 #define USART3_TX 8

4.16.2 Typedef Documentation

4.16.2.1 typedef struct Buffer Buffer

4.16.3 Function Documentation

4.16.3.1 void initGPIOUsart3 ( )

Launchs the GPIO ports needed by Usart to work.

Returns

void.

4.16.3.2 void initUsart3 (uint32_t *baudRate,* uint32_t *interrupzioak*)

Initializes Usart 3 with a given baudrate and optional configuration for interruption.

Parameters

baudrate	uint32_t type specifies the baudrate of Usart. interrupzioak: uint32_t acts as a boolean to define whether is needed interrupion configuration.

Returns

void.

4.16.3.3 void ourUSART3Handler (void)

4.16.3.4 uint8_t popBuffer (Buffer ∗ *buffer*)

Pops the first value of the given buffer.

Parameters

buffer	Buffer pointer type.

Returns

pop: the first value of the buffer.

4.16.3.5 void pushBuffer (Buffer ∗ *buffer,* uint8_t *balioa*)

Pushes a new value to the buffer array.

Parameters

buffer	Buffer pointer type.
balioa	uint8_t type the value that needs to be pushed into the array

4.16.3.6 uint32_t readBufferSize (void)

4.16.3.7 uint32_t readUart3 (uint8_t ∗ *pMsg,* uint32_t *maxLen*)

4.16.3.8 void writeByte (uint8_t *mezua*)

Writes one byte on data register (DR) mezua: uint8_t the message that needs to be written in the data register.

Returns

void.

4.16.3.9 void writeUart3 (uint8_t ∗ *mezua,* uint32_t *luzera*)

/ Copy the message to the buffer when it can.

Parameters

mezua	uint8_t pointer holds the message value.
luzera	uint32_t type the size of the message.

Returns

void.

4.16.3.10 void writeUart3Blocking (uint8_t ∗ *mezua,* uint32_t *luzera*)

4.16.4 Variable Documentation

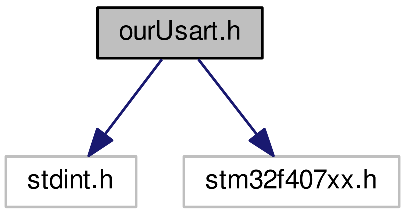
4.16.4.1 Buffer bufferIdatzi

4.16.4.2 Buffer bufferIrakurri

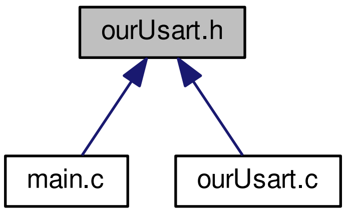
4.17 ourUsart.h File Reference

```
#include <stdint.h>
#include <stm32f407xx.h>
```

Include dependency graph for ourUsart.h:



This graph shows which files directly or indirectly include this file:



Functions

- void initUsart3 (uint32_t baudRate, uint32_t interrupzioak)
- uint32_t readBufferSize (void)
- void writeUart3 (uint8_t ∗mezua, uint32_t luzera)
- uint32_t readUart3 (uint8_t ∗pMsg, uint32_t maxLen)
- void writeUart3Blocking (uint8_t ∗mezua, uint32_t luzera)
- void writeByte (uint8_t mezua)
- void ourUSART3Handler (void)
- void writeToUart (uint8_t ∗pMsg)

4.17.1 Function Documentation

4.17.1.1 void initUsart3 (uint32_t *baudRate,* uint32_t *interrupzioak*)

Initializes Usart 3 with a given baudrate and optional configuration for interruption.

Parameters

baudrate	uint32_t type specifies the baudrate of Usart. interrupzioak: uint32_t acts as a boolean to define whether is needed interrupion configuration.

Returns

void.

4.17.1.2 void ourUSART3Handler (void)

4.17.1.3 uint32_t readBufferSize (void)

4.17.1.4 uint32_t readUart3 (uint8_t ∗ *pMsg,* uint32_t *maxLen*)

4.17.1.5 void writeByte (uint8_t *mezua*)

Writes one byte on data register (DR) mezua: uint8_t the message that needs to be written in the data register.

Returns

void.

4.17.1.6 void writeToUart (uint8_t ∗ *pMsg*)

4.17.1.7 void writeUart3 (uint8_t ∗ *mezua,* uint32_t *luzera*)

/ Copy the message to the buffer when it can.

Parameters

mezua	uint8_t pointer holds the message value.
luzera	uint32_t type the size of the message.

Returns

void.

4.17.1.8 void writeUart3Blocking (uint8_t ∗ *mezua,* uint32_t *luzera*)