# CarRunner Game

Ver1.0

Generated by Doxygen 1.8.16

# Chapter 1

# Module Index

## 1.1 Modules

Here is a list of all modules:

# Chapter 2

# Data Structure Index

## 2.1 Data Structures

Here are the data structures with brief descriptions:

# Chapter 3

# File Index

## 3.1 File List

Here is a list of all files with brief descriptions:

# Chapter 4

# Module Documentation

## 4.1 BUTTONS

Buttons mapping.

### Macros

- #define **L** 1
- #define **R** 2
- #define **S** 4

### 4.1.1 Detailed Description

Buttons mapping.

### 4.1.2 Macro Definition Documentation

#### 4.1.2.1 L

```
#define L 1
```

Definition at line 15 of file CarRunner.h.

#### 4.1.2.2 R

```
#define R 2
```

Definition at line 16 of file CarRunner.h.

#### 4.1.2.3 S

```
#define S 4
```

Definition at line 17 of file CarRunner.h.

## 4.2 ADXL345_DEVID

### Macros

- #define **ADXL345_DEVID_RESET_VALUE** (0x00E5)
- #define **ADXL345_REG_DEVID** (0x00)

### 4.2.1 Detailed Description

### 4.2.2 Macro Definition Documentation

#### 4.2.2.1 ADXL345_DEVID_RESET_VALUE

```
#define ADXL345_DEVID_RESET_VALUE (0x00E5)
```

Definition at line 19 of file ADXL345.h.

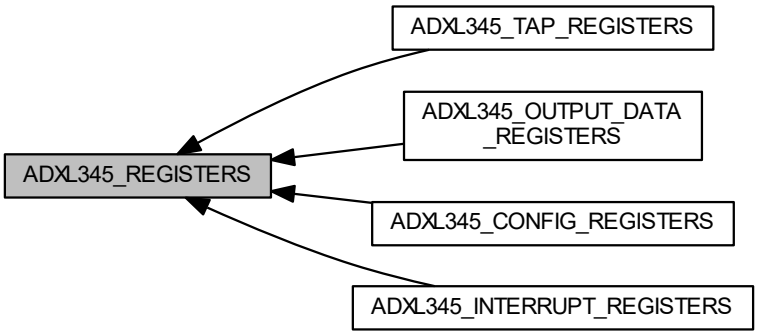
#### 4.2.2.2 ADXL345_REG_DEVID

```
#define ADXL345_REG_DEVID (0x00)
```

Definition at line 20 of file ADXL345.h.

## 4.3 ADXL345_REGISTERS

Collaboration diagram for ADXL345_REGISTERS:



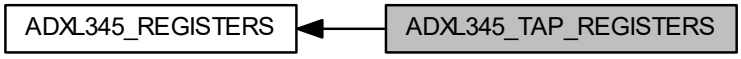

### Modules

- **ADXL345_TAP_REGISTERS**
- **ADXL345_CONFIG_REGISTERS**
- **ADXL345_INTERRUPT_REGISTERS**
- **ADXL345_OUTPUT_DATA_REGISTERS**

### 4.3.1 Detailed Description

## 4.4 ADXL345_TAP_REGISTERS

Collaboration diagram for ADXL345_TAP_REGISTERS:




### Macros

- #define **ADXL345_REG_THRESH_TAP** (0x1D)
- #define **ADXL345_REG_DUR** (0x21)
- #define **ADXL345_REG_LATENT** (0x22)
- #define **ADXL345_REG_WINDOW** (0x23)
- #define **ADXL345_REG_THRESH_ACT** (0x24)
- #define **ADXL345_REG_TAP_AXES** (0x2A)
- #define **ADXL345_REG_ACT_TAP_STATUS** (0x2B)

### 4.4.1 Detailed Description

### 4.4.2 Macro Definition Documentation

#### 4.4.2.1 ADXL345_REG_ACT_TAP_STATUS

```
#define ADXL345_REG_ACT_TAP_STATUS (0x2B)
```

Definition at line 38 of file ADXL345.h.

#### 4.4.2.2 ADXL345_REG_DUR

```
#define ADXL345_REG_DUR (0x21)
```

Definition at line 33 of file ADXL345.h.

#### 4.4.2.3 ADXL345_REG_LATENT

```
#define ADXL345_REG_LATENT (0x22)
```

Definition at line 34 of file ADXL345.h.

#### 4.4.2.4 ADXL345_REG_TAP_AXES

```
#define ADXL345_REG_TAP_AXES (0x2A)
```

Definition at line 37 of file ADXL345.h.

#### 4.4.2.5 ADXL345_REG_THRESH_ACT

```
#define ADXL345_REG_THRESH_ACT (0x24)
```

Definition at line 36 of file ADXL345.h.

#### 4.4.2.6 ADXL345_REG_THRESH_TAP

```
#define ADXL345_REG_THRESH_TAP (0x1D)
```

Definition at line 32 of file ADXL345.h.

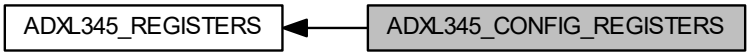
#### 4.4.2.7 ADXL345_REG_WINDOW

```
#define ADXL345_REG_WINDOW (0x23)
```

Definition at line 35 of file ADXL345.h.

## 4.5 ADXL345_CONFIG_REGISTERS

Collaboration diagram for ADXL345_CONFIG_REGISTERS:




### Macros

- #define **ADXL345_REG_BW_RATE** (0x2C)
- #define **ADXL345_REG_POWER_CTL** (0x2D)

### 4.5.1 Detailed Description

### 4.5.2 Macro Definition Documentation

#### 4.5.2.1 ADXL345_REG_BW_RATE

```
#define ADXL345_REG_BW_RATE (0x2C)
```

Definition at line 46 of file ADXL345.h.

#### 4.5.2.2 ADXL345_REG_POWER_CTL

```
#define ADXL345_REG_POWER_CTL (0x2D)
```

Definition at line 47 of file ADXL345.h.

## 4.6 ADXL345_INTERRUPT_REGISTERS

Collaboration diagram for ADXL345_INTERRUPT_REGISTERS:


### Macros

- #define **ADXL345_REG_INT_ENABLE** (0x2E)
- #define **ADXL345_REG_INT_MAP** (0x2F)
- #define **ADXL345_REG_INT_SOURCE** (0x30)

### 4.6.1 Detailed Description

### 4.6.2 Macro Definition Documentation

#### 4.6.2.1 ADXL345_REG_INT_ENABLE

```
#define ADXL345_REG_INT_ENABLE (0x2E)
```

Definition at line 55 of file ADXL345.h.

#### 4.6.2.2 ADXL345_REG_INT_MAP

```
#define ADXL345_REG_INT_MAP (0x2F)
```

Definition at line 56 of file ADXL345.h.

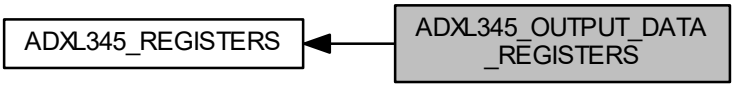
#### 4.6.2.3 ADXL345_REG_INT_SOURCE

```
#define ADXL345_REG_INT_SOURCE (0x30)
```

Definition at line 57 of file ADXL345.h.

## 4.7 ADXL345_OUTPUT_DATA_REGISTERS

Collaboration diagram for ADXL345_OUTPUT_DATA_REGISTERS:




### Macros

- #define **ADXL345_REG_DATA_FORMAT** (0x31)
- #define **ADXL345_REG_DATAX0** (0x32)
- #define **ADXL345_REG_DATAX1** (0x33)
- #define **ADXL345_REG_DATAY0** (0x34)
- #define **ADXL345_REG_DATAY1** (0x35)
- #define **ADXL345_REG_DATAZ0** (0x36)
- #define **ADXL345_REG_DATAZ1** (0x37)

### 4.7.1 Detailed Description

### 4.7.2 Macro Definition Documentation

#### 4.7.2.1 ADXL345_REG_DATA_FORMAT

```
#define ADXL345_REG_DATA_FORMAT (0x31)
```

Definition at line 65 of file ADXL345.h.

#### 4.7.2.2 ADXL345_REG_DATAX0

```
#define ADXL345_REG_DATAX0 (0x32)
```

Definition at line 66 of file ADXL345.h.

#### 4.7.2.3 ADXL345_REG_DATAX1

```
#define ADXL345_REG_DATAX1 (0x33)
```

Definition at line 67 of file ADXL345.h.

#### 4.7.2.4 ADXL345_REG_DATAY0

```
#define ADXL345_REG_DATAY0 (0x34)
```

Definition at line 68 of file ADXL345.h.

#### 4.7.2.5 ADXL345_REG_DATAY1

```
#define ADXL345_REG_DATAY1 (0x35)
```

Definition at line 69 of file ADXL345.h.

#### 4.7.2.6 ADXL345_REG_DATAZ0

```
#define ADXL345_REG_DATAZ0 (0x36)
```

Definition at line 70 of file ADXL345.h.

#### 4.7.2.7 ADXL345_REG_DATAZ1

```
#define ADXL345_REG_DATAZ1 (0x37)
```

Definition at line 71 of file ADXL345.h.

## 4.8 GRAVITY_CONSTANTS

### Macros

- #define **ADXL345_GRAVITY_SUN** 273.95f
- #define **ADXL345_GRAVITY_EARTH** 9.80665f
- #define **ADXL345_GRAVITY_MOON** 1.622f
- #define **ADXL345_GRAVITY_MARS** 3.69f
- #define **ADXL345_GRAVITY_NONE** 1.00f

### 4.8.1 Detailed Description

### 4.8.2 Macro Definition Documentation

#### 4.8.2.1 ADXL345_GRAVITY_EARTH

```
#define ADXL345_GRAVITY_EARTH 9.80665f
```

Definition at line 84 of file ADXL345.h.

#### 4.8.2.2 ADXL345_GRAVITY_MARS

```
#define ADXL345_GRAVITY_MARS 3.69f
```

Definition at line 86 of file ADXL345.h.

#### 4.8.2.3 ADXL345_GRAVITY_MOON

```
#define ADXL345_GRAVITY_MOON 1.622f
```

Definition at line 85 of file ADXL345.h.

#### 4.8.2.4 ADXL345_GRAVITY_NONE

```
#define ADXL345_GRAVITY_NONE 1.00f
```

Definition at line 87 of file ADXL345.h.

#### 4.8.2.5 ADXL345_GRAVITY_SUN

```
#define ADXL345_GRAVITY_SUN 273.95f
```

Definition at line 83 of file ADXL345.h.

# 4.9 IAP_RETURN_CODES

## Macros

- #define **IAP_CMD_SUCESS** 0
- #define **IAP_INVALID_COMMAND** 1
- #define **IAP_SRC_ADDR_ERROR** 2
- #define **IAP_DST_ADDR_ERROR** 3
- #define **IAP_SRC_ADDR_NOT_MAPPED** 4
- #define **IAP_DST_ADDR_NOT_MAPPED** 5
- #define **IAP_COUNT_ERROR** 6
- #define **IAP_INVALID_SECTOR** 7
- #define **IAP_SECTOR_NOT_BLANK** 8
- #define **IAP_SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION** 9
- #define **IAP_COMPARE_ERROR** 10
- #define **IAP_BUSY** 11

### 4.9.1 Detailed Description

### 4.9.2 Macro Definition Documentation

#### 4.9.2.1 IAP_BUSY

```
#define IAP_BUSY 11
```

Definition at line 27 of file Flash.h.

#### 4.9.2.2 IAP_CMD_SUCESS

```
#define IAP_CMD_SUCESS 0
```

Definition at line 16 of file Flash.h.

#### 4.9.2.3 IAP_COMPARE_ERROR

```
#define IAP_COMPARE_ERROR 10
```

Definition at line 26 of file Flash.h.

#### 4.9.2.4 IAP_COUNT_ERROR

```
#define IAP_COUNT_ERROR 6
```

Definition at line 22 of file Flash.h.

#### 4.9.2.5 IAP_DST_ADDR_ERROR

```
#define IAP_DST_ADDR_ERROR 3
```

Definition at line 19 of file Flash.h.

#### 4.9.2.6 IAP_DST_ADDR_NOT_MAPPED

```
#define IAP_DST_ADDR_NOT_MAPPED 5
```

Definition at line 21 of file Flash.h.

#### 4.9.2.7 IAP_INVALID_COMMAND

```
#define IAP_INVALID_COMMAND 1
```

Definition at line 17 of file Flash.h.

#### 4.9.2.8 IAP_INVALID_SECTOR

```
#define IAP_INVALID_SECTOR 7
```

Definition at line 23 of file Flash.h.

#### 4.9.2.9 IAP_SECTOR_NOT_BLANK

```
#define IAP_SECTOR_NOT_BLANK 8
```

Definition at line 24 of file Flash.h.

#### 4.9.2.10 IAP_SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION

```
#define IAP_SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION 9
```

Definition at line 25 of file Flash.h.

#### 4.9.2.11 IAP_SRC_ADDR_ERROR

```
#define IAP_SRC_ADDR_ERROR 2
```

Definition at line 18 of file Flash.h.

#### 4.9.2.12 IAP_SRC_ADDR_NOT_MAPPED

```
#define IAP_SRC_ADDR_NOT_MAPPED 4
```

Definition at line 20 of file Flash.h.

# 4.10 DISPLAY_DIMENSIONS

## Macros

- #define **LCDText_LINES** 2
- #define **LCDText_COLUMNS** 16

## 4.10.1 Detailed Description

## 4.10.2 Macro Definition Documentation

### 4.10.2.1 LCDText_COLUMNS

```
#define LCDText_COLUMNS 16
```

Definition at line 17 of file lcd.h.

### 4.10.2.2 LCDText_LINES

```
#define LCDText_LINES 2
```

Definition at line 16 of file lcd.h.

# 4.11 DISPLAY_COMMANDS

## Macros

- #define **LCDText_CMD_DISPLAY_OFF** 0x08
- #define **LCDText_CMD_DISPLAY_ON** 0x0C
- #define **LCDText_CMD_DISPLAY_CLEAR** 1
- #define **LCDText_CMD_ENTRY_MODE_SET** 0x06
- #define **LCDText_CMD_FUNCTION_SET** 0x28
- #define **LCDText_CMD_RETURN_HOME** 2
- #define **LCDText_CMD_SET_DDRAM_ADDR** 0x80

### 4.11.1 Detailed Description

### 4.11.2 Macro Definition Documentation

#### 4.11.2.1 LCDText_CMD_DISPLAY_CLEAR

```
#define LCDText_CMD_DISPLAY_CLEAR 1
```

Definition at line 35 of file lcd.h.

#### 4.11.2.2 LCDText_CMD_DISPLAY_OFF

```
#define LCDText_CMD_DISPLAY_OFF 0x08
```

Definition at line 33 of file lcd.h.

#### 4.11.2.3 LCDText_CMD_DISPLAY_ON

```
#define LCDText_CMD_DISPLAY_ON 0x0C
```

Definition at line 34 of file lcd.h.

#### 4.11.2.4 LCDText_CMD_ENTRY_MODE_SET

```
#define LCDText_CMD_ENTRY_MODE_SET 0x06
```

Definition at line 36 of file lcd.h.

#### 4.11.2.5 LCDText_CMD_FUNCTION_SET

```
#define LCDText_CMD_FUNCTION_SET 0x28
```

Definition at line 37 of file lcd.h.

#### 4.11.2.6 LCDText_CMD_RETURN_HOME

```
#define LCDText_CMD_RETURN_HOME 2
```

Definition at line 38 of file lcd.h.

#### 4.11.2.7 LCDText_CMD_SET_DDRAM_ADDR

```
#define LCDText_CMD_SET_DDRAM_ADDR 0x80
```

Definition at line 39 of file lcd.h.

# 4.12 WAIT

This package provides the core capabilities for wait functions.

Collaboration diagram for WAIT:


## Modules

- **WAIT Public Functions**

### 4.12.1 Detailed Description

This package provides the core capabilities for wait functions.

## 4.13 WAIT Public Functions

Collaboration diagram for WAIT Public Functions:


### Functions

- int32_t **WAIT_Init** (void)

  *Initialises the wait API for 1 ms resolution.*
- void **WAIT_Milliseconds** (uint32_t millis)

  *Waits a number of milliseconds.*
- uint32_t **WAIT_GetElapsedMillis** (uint32_t start)

  *Get difference in milliseconds from parameter.*
- void **WAIT_ChronoUs** (uint32_t waitUs)

  *Waits waitUs microseconds.*

### 4.13.1 Detailed Description

### 4.13.2 Function Documentation

#### 4.13.2.1 WAIT_ChronoUs()

```
void WAIT_ChronoUs (
            uint32_t waitUs )
```

Waits waitUs microseconds.

**Parameters**

| | |
|---|---|
| *waitUs* | miliseconds to wait |

Definition at line 50 of file wait.c.

Referenced by BUTTON_Hit(), gameRoutine(), LCDText_Clear(), LCDText_CreateChar(), LCDText_Init(), LCD↩ Text_WriteChar(), LCDText_WriteCmd(), LCDText_WriteNibble(), LCDText_WriteString(), and playerScoresShow↩ Down().

#### 4.13.2.2 WAIT_GetElapsedMillis()

```
uint32_t WAIT_GetElapsedMillis (
             uint32_t start )
```

Get difference in milliseconds from parameter.

**Parameters**

| | |
|---|---|
| *start* | if 0 get current milliseconds. |

Definition at line 45 of file wait.c.

Referenced by gameRoutine(), main(), and saveUser().

#### 4.13.2.3 WAIT_Init()

```
int32_t WAIT_Init (
             void  )
```

Initialises the wait API for 1 ms resolution.

**Returns**

0 if initialisation successed; -1 if fails.

**Note**

This function must be called prior to any other WAIT functions, and use the SYSTICK resource.

Definition at line 32 of file wait.c.

References SYSTICK_FREQ.

Referenced by main().

#### 4.13.2.4 WAIT_Milliseconds()

```
void WAIT_Milliseconds (
             uint32_t millis )
```

Waits a number of milliseconds.

**Parameters**

| | |
|---|---|
| *millis* | the whole number of milliseconds to wait. |

Definition at line 24 of file wait.c.

# 4.14 button gpio definition

## Macros

- #define **BUTTON_ONE** 24
- #define **BUTTON_TWO** 25
- #define **BUTTON_THREE** 26

### 4.14.1 Detailed Description

### 4.14.2 Macro Definition Documentation

#### 4.14.2.1 BUTTON_ONE

```
#define BUTTON_ONE 24
```

Definition at line 19 of file button.c.

#### 4.14.2.2 BUTTON_THREE

```
#define BUTTON_THREE 26
```

Definition at line 21 of file button.c.

#### 4.14.2.3 BUTTON_TWO

```
#define BUTTON_TWO 25
```

Definition at line 20 of file button.c.

# Chapter 5

# Data Structure Documentation

## 5.1 key_state Struct Reference

`#include <button.h>`

### Data Fields

- int **key**
- int **state**

### 5.1.1 Detailed Description

Definition at line 17 of file button.h.

### 5.1.2 Field Documentation

#### 5.1.2.1 key

`int key_state::key`

Definition at line 18 of file button.h.

Referenced by BUTTON_Hit(), gameRoutine(), main(), Menu(), playerScoresShowDown(), saveUser(), and Time↩
_changeRoutine().

#### 5.1.2.2 state

`int key_state::state`

Definition at line 19 of file button.h.

Referenced by BUTTON_Hit(), gameRoutine(), main(), Menu(), playerScoresShowDown(), saveUser(), and Time↩
_changeRoutine().

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

- C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/SE2021/inc/ **button.h**

# Chapter 6

# File Documentation

## 6.1 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/CarRunner/inc/CarRunner.h File Reference

### Macros

- #define **L** 1
- #define **R** 2
- #define **S** 4

## 6.2 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/CarRunner/inc/game.h File Reference

### Functions

- void **gameInit** ()

  *initializes the module*
- void **gameRoutine** (void)

  *routine that interfaces with buttons, lcd and accelerometer to play the game. It also saves scores and player names.*

### 6.2.1 Function Documentation

#### 6.2.1.1 gameInit()

```
void gameInit ( )
```

initializes the module

Definition at line 30 of file game.c.

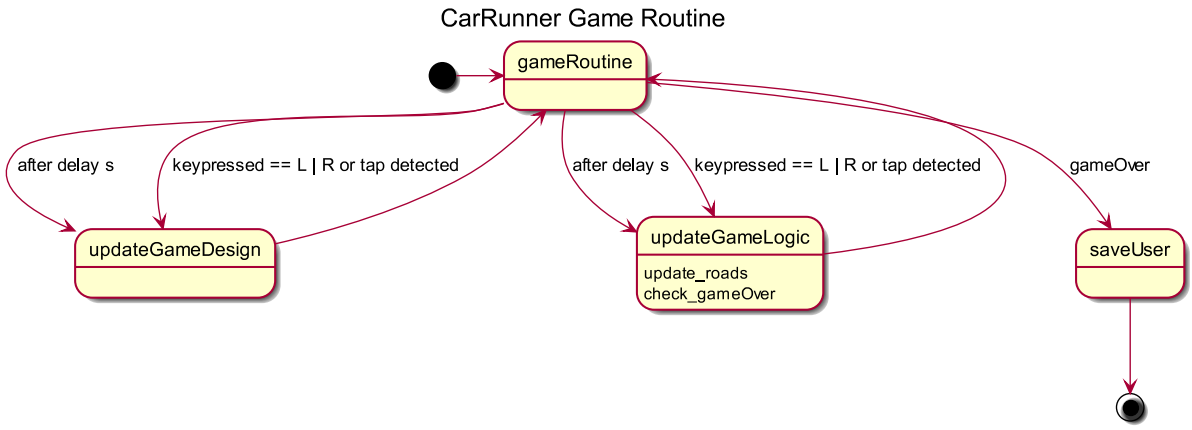
References car, and LCDText_CreateChar().

Referenced by main().

#### 6.2.1.2 gameRoutine()

```
void gameRoutine (
             void  )
```

routine that interfaces with buttons, lcd and accelerometer to play the game. It also saves scores and player names.




Definition at line 56 of file game.c.

References ADXL345_isTap(), BUTTON_GetButtonsEvents(), car, delay, initValues(), key_state::key, L, LCDText↩ _Clear(), LCDText_Locate(), LCDText_Printf(), LCDText_WriteString(), points, PRESSED, R, readScores(), save↩ User(), key_state::state, updateGameDesign(), updateGameLogic(), WAIT_ChronoUs(), and WAIT_GetElapsed↩ Millis().

Referenced by main().

## 6.3 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/Car↩ Runner/inc/saver.h File Reference

### Functions

- void **saveScore** (int score, char ∗name, int name_size)
- int ∗ **readScores** ()

  *reads from flash all the scores*
- int **listSize** ()
- void **readNames** (char ∗strings[ ])

  *function that reads from flash all the names its required that strings is started with size[listSize][17] to accommodate for max size names*

### 6.3.1 Function Documentation

#### 6.3.1.1 listSize()

```
int listSize ( )
```

**Returns**

the current number of players that exist

Definition at line 83 of file saver.c.

References checkClean(), and names.

Referenced by playerScoresShowDown().

#### 6.3.1.2 readNames()

```
void readNames (
            char * strings[] )
```

function that reads from flash all the names its required that strings is started with size[listSize][17] to accommodate for max size names

Definition at line 89 of file saver.c.

References names.

Referenced by playerScoresShowDown().

#### 6.3.1.3 readScores()

```
int* readScores ( )
```

reads from flash all the scores

**Returns**

array of scores

Definition at line 70 of file saver.c.

References checkClean(), and scores.

Referenced by gameRoutine(), and playerScoresShowDown().

#### 6.3.1.4 saveScore()

```
void saveScore (
            int score,
            char * name,
            int name_size )
```

Definition at line 22 of file saver.c.

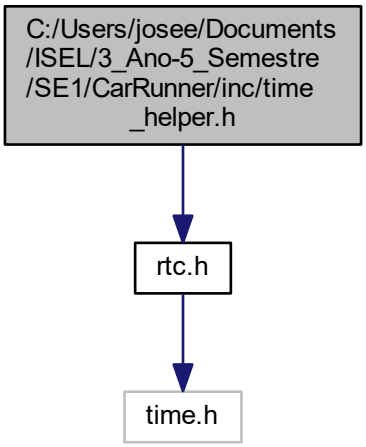
References checkClean(), FLASH_WriteArray(), IAP_CMD_SUCESS, names, and scores.

Referenced by saveUser().

## 6.4 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/CarRunner/inc/time_helper.h File Reference

```
#include "rtc.h"
```

Include dependency graph for time_helper.h:




### Functions

- void **Time_changeRoutine** (time_t seconds)

  *routine that interfaces with lcd and buttons to change time in rtc*
- void **update_DateTimeDisplay** (struct tm datetime)

  *updates lcd with date and time*

### 6.4.1 Function Documentation

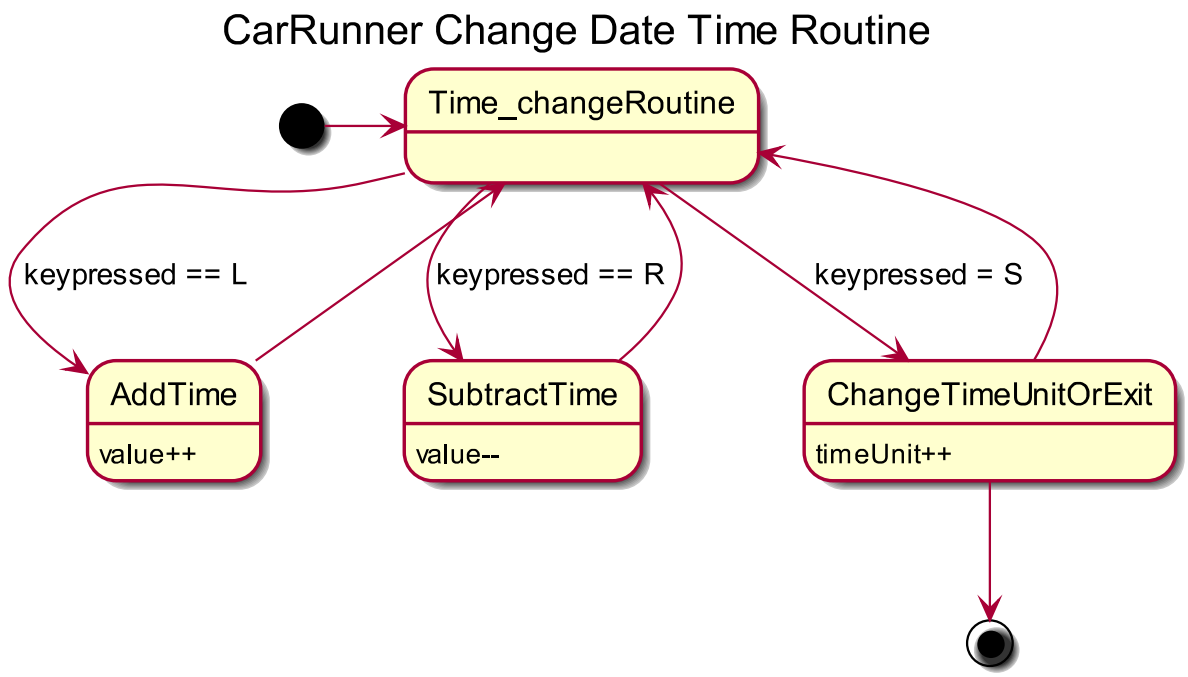
#### 6.4.1.1 Time_changeRoutine()

```
void Time_changeRoutine (
            time_t seconds )
```

routine that interfaces with lcd and buttons to change time in rtc

**Parameters**

| | |
|---|---|
| *seconds* | starting time |




Definition at line 30 of file time_helper.c.

References BUTTON_GetButtonsEvents(), key_state::key, LCDText_Clear(), LCDText_CursorOff(), LCDText↩_CursorOn(), LCDText_Locate(), LCDText_WriteString(), PRESSED, RTC_SetValue(), key_state::state, and update_DateTimeDisplay().

Referenced by routineChooser().

#### 6.4.1.2 update_DateTimeDisplay()

```
void update_DateTimeDisplay (
            struct tm datetime )
```

updates lcd with date and time

**Parameters**

| | |
|---|---|
| *datetime* | time to be displayed |

Definition at line 85 of file time_helper.c.

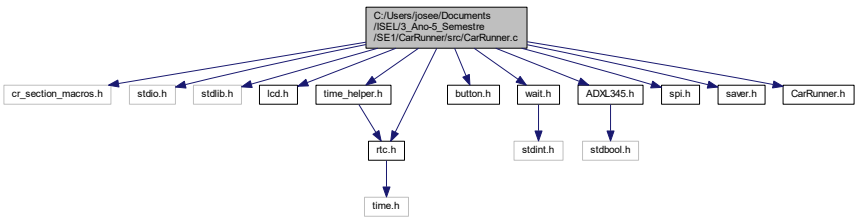
References LCDText_Locate(), and LCDText_WriteString().

Referenced by main(), and Time_changeRoutine().

## 6.5 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/Car↩ Runner/src/CarRunner.c File Reference

```
#include <cr_section_macros.h>
#include <stdio.h>
#include <stdlib.h>
#include "lcd.h"
#include "time_helper.h"
#include "button.h"
#include "rtc.h"
#include "wait.h"
#include "ADXL345.h"
#include "spi.h"
#include "saver.h"
#include "CarRunner.h"
```

Include dependency graph for CarRunner.c:




### Macros

- #define **PRESSING_TIME** 2000

### Functions

- int **main** (void)
- void **Menu** ()
- void **routineChooser** (int routine)
- void **playerScoresShowDown** ()

### Variables

- char * **menu_options** [3] = {"Change Time","Show Players\nScores", "Exit"}

## 6.5.1 Macro Definition Documentation

### 6.5.1.1 PRESSING_TIME

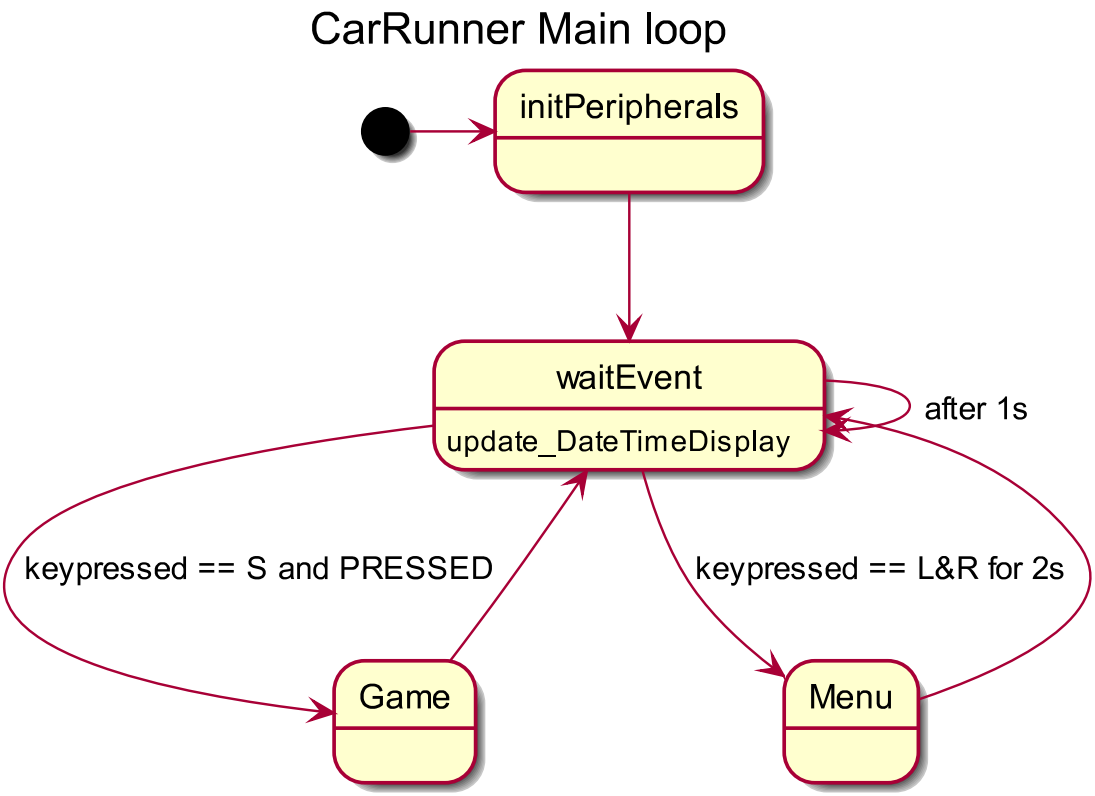
```
#define PRESSING_TIME 2000
```

Definition at line 30 of file CarRunner.c.

## 6.5.2 Function Documentation

### 6.5.2.1 main()

```
int main (
            void  )
```

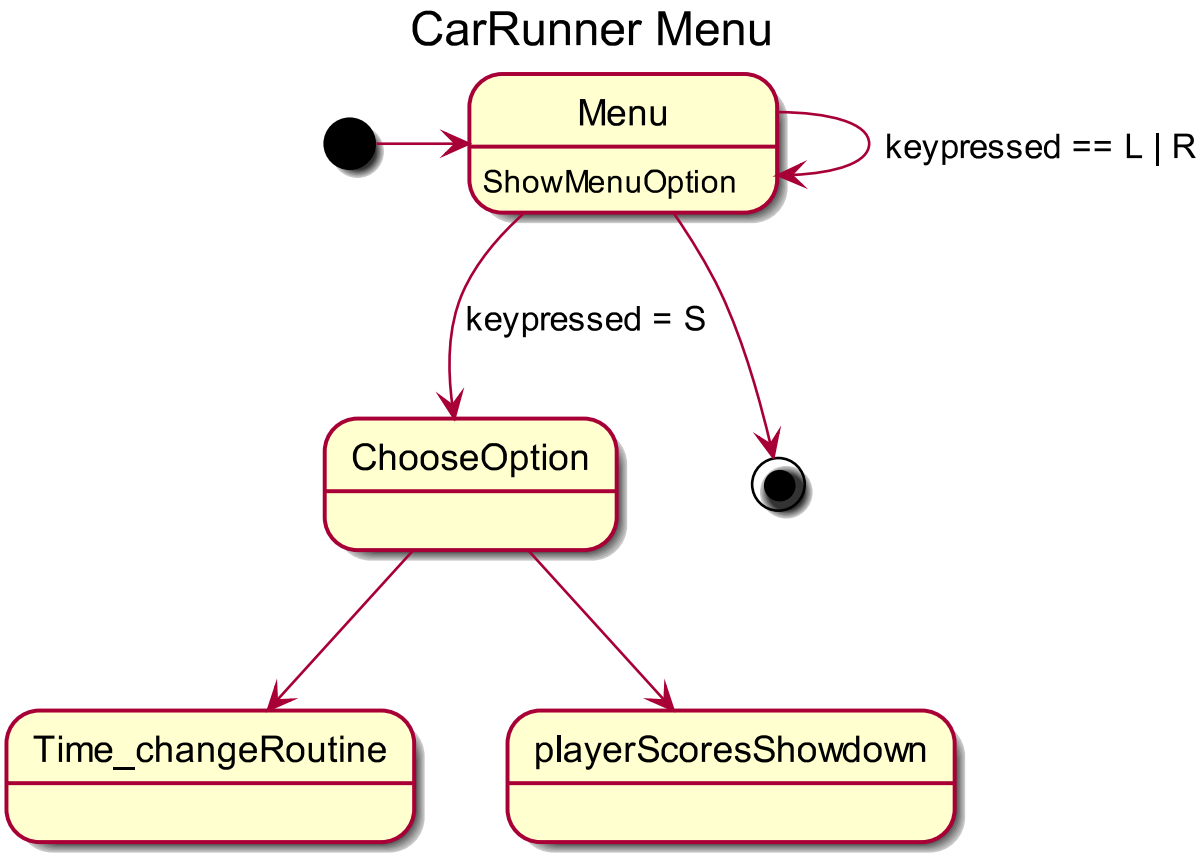



Definition at line 47 of file CarRunner.c.

References ADXL345_Init(), BUTTON_GetButtonsEvents(), BUTTON_Init(), gameInit(), gameRoutine(), key_↩ state::key, keyState, L, LCDText_Init(), Menu(), PRESSED, PRESSING_TIME, R, REPEATED, RTC_GetValue(), RTC_Init(), S, SPI_Init(), key_state::state, update_DateTimeDisplay(), WAIT_GetElapsedMillis(), and WAIT_Init().

#### 6.5.2.2 Menu()

```
void Menu ( )
```



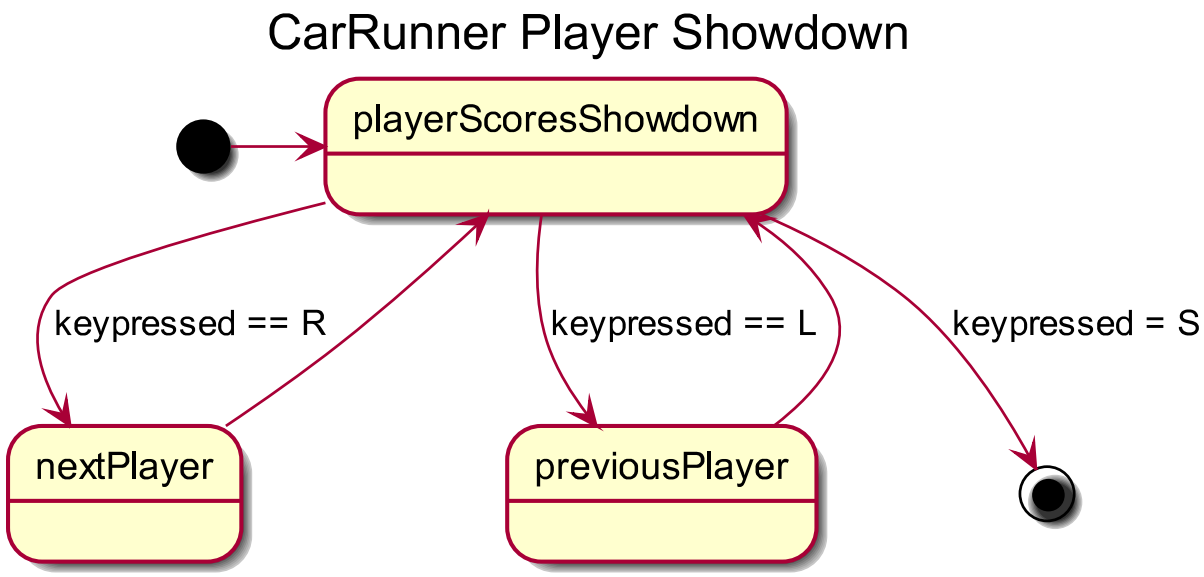

Definition at line 109 of file CarRunner.c.

References BUTTON_GetButtonsEvents(), key_state::key, keyState, L, LCDText_Clear(), LCDText_WriteString(), menu_options, PRESSED, R, routineChooser(), S, and key_state::state.

Referenced by main().

#### 6.5.2.3 playerScoresShowDown()

```
void playerScoresShowDown ( )
```




Definition at line 176 of file CarRunner.c.

References BUTTON_GetButtonsEvents(), key_state::key, keyState, L, LCDText_Clear(), LCDText_Printf(), LCD↩ Text_WriteString(), listSize(), PRESSED, R, readNames(), readScores(), S, scores, key_state::state, and WAIT_↩ ChronoUs().

Referenced by routineChooser().

#### 6.5.2.4 routineChooser()

```
void routineChooser (
            int routine )
```

Definition at line 150 of file CarRunner.c.

References playerScoresShowDown(), RTC_GetSeconds(), and Time_changeRoutine().

Referenced by Menu().

### 6.5.3 Variable Documentation

#### 6.5.3.1 menu_options

```
char* menu_options[3] = {"Change Time","Show Players\nScores", "Exit"}
```

Definition at line 93 of file CarRunner.c.

Referenced by Menu().

## 6.6 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/CarRunner/src/cr_startup_lpc175x_6x.c File Reference

### Macros

- #define **WEAK** **__attribute__** ((weak))
- #define **ALIAS**(f) **__attribute__** ((weak, alias (#f)))

### Functions

- void **ResetISR** (void)
- **WEAK** void **NMI_Handler** (void)
- **WEAK** void **HardFault_Handler** (void)
- **WEAK** void **MemManage_Handler** (void)
- **WEAK** void **BusFault_Handler** (void)
- **WEAK** void **UsageFault_Handler** (void)
- **WEAK** void **SVC_Handler** (void)
- **WEAK** void **DebugMon_Handler** (void)
- **WEAK** void **PendSV_Handler** (void)
- **WEAK** void **SysTick_Handler** (void)
- **WEAK** void **IntDefaultHandler** (void)
- void **WDT_IRQHandler** (void TIMER0_IRQHandler() **ALIAS**( **IntDefaultHandler**) void)
- **__attribute__** ((section(".after_vectors")))

### Variables

- unsigned int **__data_section_table**
- unsigned int **__data_section_table_end**
- unsigned int **__bss_section_table**
- unsigned int **__bss_section_table_end**

### 6.6.1 Macro Definition Documentation

#### 6.6.1.1 ALIAS

```
#define ALIAS(
              f )  __attribute__ ((weak, alias (#f)))
```

Definition at line 37 of file cr_startup_lpc175x_6x.c.

#### 6.6.1.2 WEAK

```
#define WEAK  __attribute__ ((weak))
```

Definition at line 36 of file cr_startup_lpc175x_6x.c.

### 6.6.2 Function Documentation

#### 6.6.2.1 __attribute__()

```
__attribute__ (
             (section(".after_vectors"))  )
```

Definition at line 221 of file cr_startup_lpc175x_6x.c.

#### 6.6.2.2 BusFault_Handler()

```
WEAK void BusFault_Handler (
             void  )
```

#### 6.6.2.3 DebugMon_Handler()

```
WEAK void DebugMon_Handler (
             void  )
```

#### 6.6.2.4 HardFault_Handler()

```
WEAK void HardFault_Handler (
             void  )
```

#### 6.6.2.5 IntDefaultHandler()

```
WEAK void IntDefaultHandler (
             void  )
```

#### 6.6.2.6 MemManage_Handler()

```
WEAK void MemManage_Handler (
            void  )
```

#### 6.6.2.7 NMI_Handler()

```
WEAK void NMI_Handler (
            void  )
```

#### 6.6.2.8 PendSV_Handler()

```
WEAK void PendSV_Handler (
            void  )
```

#### 6.6.2.9 ResetISR()

```
void ResetISR (
            void  )
```

#### 6.6.2.10 SVC_Handler()

```
WEAK void SVC_Handler (
            void  )
```

#### 6.6.2.11 SysTick_Handler()

```
WEAK void SysTick_Handler (
            void  )
```

Definition at line 19 of file wait.c.

#### 6.6.2.12 UsageFault_Handler()

```
WEAK void UsageFault_Handler (
            void  )
```

#### 6.6.2.13 WDT_IRQHandler()

```
void WDT_IRQHandler (
            void TIMER0_IRQHandler()  ALIAS( IntDefaultHandler) void )
```

Definition at line 77 of file cr_startup_lpc175x_6x.c.

### 6.6.3 Variable Documentation

#### 6.6.3.1 __bss_section_table

```
unsigned int __bss_section_table
```

#### 6.6.3.2 __bss_section_table_end

```
unsigned int __bss_section_table_end
```

#### 6.6.3.3 __data_section_table

```
unsigned int __data_section_table
```

#### 6.6.3.4 __data_section_table_end

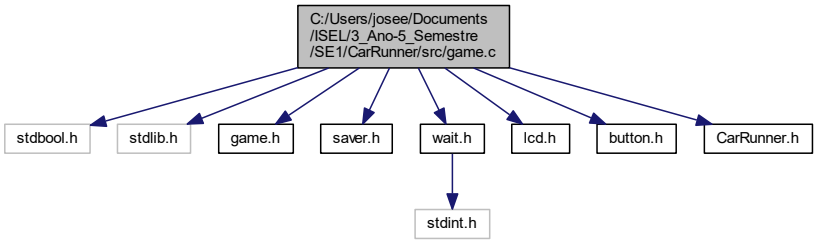
```
unsigned int __data_section_table_end
```

## 6.7 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/CarRunner/src/crp.c File Reference

## 6.8 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/CarRunner/src/game.c File Reference

```
#include <stdbool.h>
#include <stdlib.h>
#include "game.h"
#include "saver.h"
#include "wait.h"
#include "lcd.h"
#include "button.h"
#include "CarRunner.h"
```

Include dependency graph for game.c:




### Macros

- #define **CAR** 1
- #define **OBS** 2

### Functions

- void **gameInit** ()

  *initializes the module*
- void **gameRoutine** ()

  *routine that interfaces with buttons, lcd and accelerometer to play the game. It also saves scores and player names.*
- void **saveUser** (int **points**)
- void **initValues** ()
- int **updateGameLogic** ()
- void **updateGameDesign** ()

## Variables

- bool **car** = 0
- bool **upRoad** [16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
- bool **downRoad** [16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
- int **probabilityToSpawn** = 20
- int **points** = 0
- int **delay** = 1000

### 6.8.1 Macro Definition Documentation

#### 6.8.1.1 CAR

```
#define CAR 1
```

Definition at line 18 of file game.c.

#### 6.8.1.2 OBS

```
#define OBS 2
```

Definition at line 19 of file game.c.

### 6.8.2 Function Documentation

#### 6.8.2.1 gameInit()

```
void gameInit ( )
```

initializes the module

Definition at line 30 of file game.c.

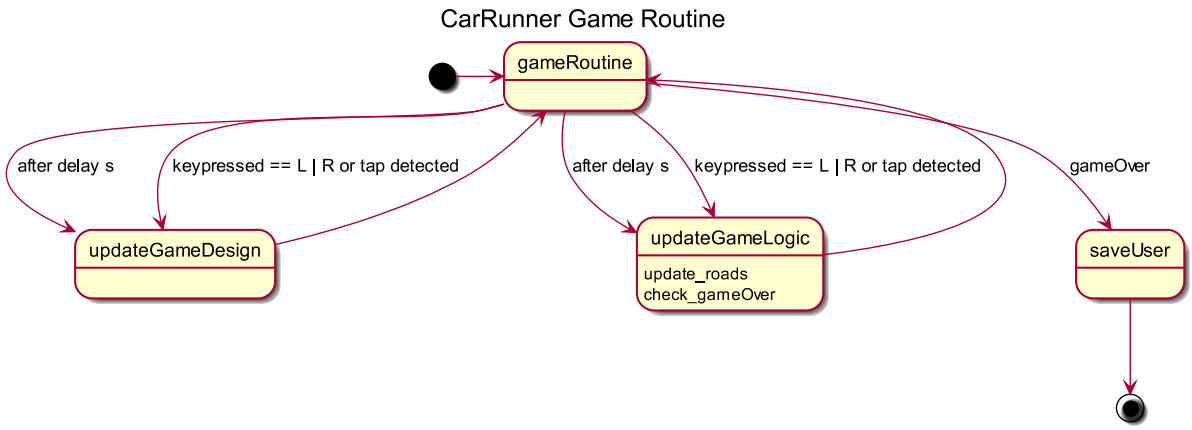
References car, and LCDText_CreateChar().

Referenced by main().

#### 6.8.2.2 gameRoutine()

```
void gameRoutine (
             void  )
```

routine that interfaces with buttons, lcd and accelerometer to play the game. It also saves scores and player names.




Definition at line 56 of file game.c.

References ADXL345_isTap(), BUTTON_GetButtonsEvents(), car, delay, initValues(), key_state::key, L, LCDText↩ _Clear(), LCDText_Locate(), LCDText_Printf(), LCDText_WriteString(), points, PRESSED, R, readScores(), save↩ User(), key_state::state, updateGameDesign(), updateGameLogic(), WAIT_ChronoUs(), and WAIT_GetElapsed↩ Millis().

Referenced by main().

#### 6.8.2.3 initValues()

```
void initValues ( )
```

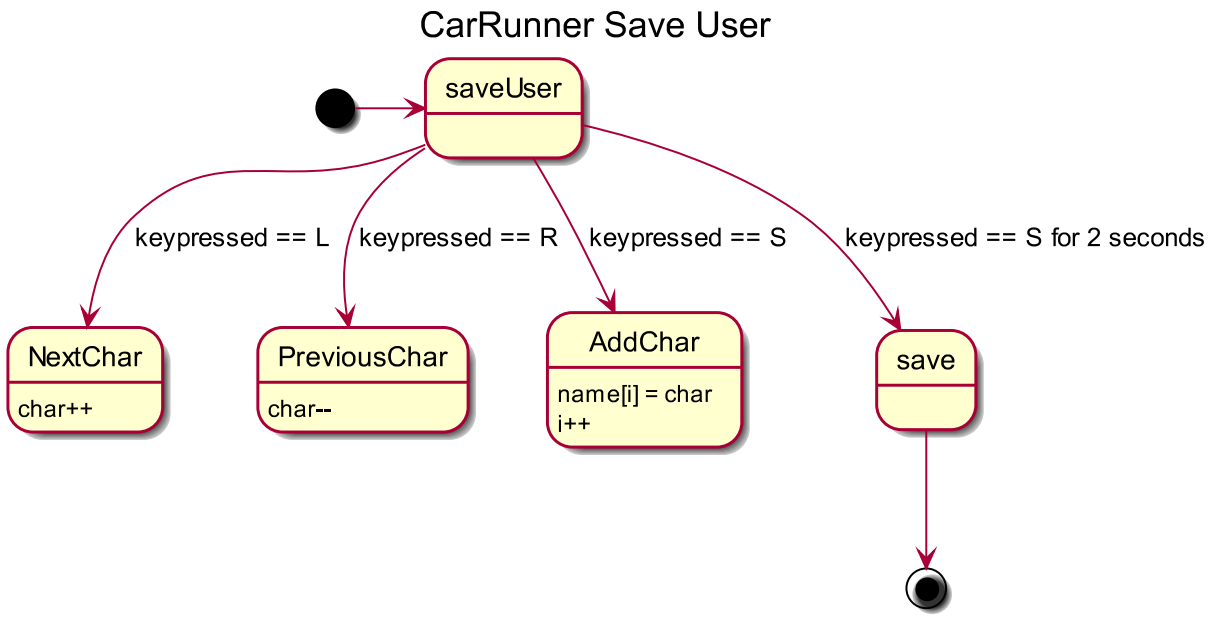
Definition at line 179 of file game.c.

References car, delay, downRoad, points, probabilityToSpawn, and upRoad.

Referenced by gameRoutine().

#### 6.8.2.4 saveUser()

```
void saveUser (
            int points )
```




Definition at line 127 of file game.c.

References BUTTON_GetButtonsEvents(), key_state::key, L, LCDText_Clear(), LCDText_CursorOff(), LCDText_↩ CursorOn(), LCDText_Locate(), LCDText_WriteChar(), LCDText_WriteString(), points, PRESSED, R, REPEATED, S, saveScore(), key_state::state, and WAIT_GetElapsedMillis().

Referenced by gameRoutine().

#### 6.8.2.5 updateGameDesign()

```
void updateGameDesign ( )
```

Definition at line 251 of file game.c.

References CAR, car, downRoad, LCDText_Locate(), LCDText_WriteString(), OBS, and upRoad.

Referenced by gameRoutine().

#### 6.8.2.6 updateGameLogic()

```
int updateGameLogic ( )
```

Definition at line 190 of file game.c.

References car, delay, downRoad, points, probabilityToSpawn, and upRoad.

Referenced by gameRoutine().

### 6.8.3 Variable Documentation

#### 6.8.3.1 car

```
bool car = 0
```

Definition at line 23 of file game.c.

Referenced by gameInit(), gameRoutine(), initValues(), updateGameDesign(), and updateGameLogic().

#### 6.8.3.2 delay

```
int delay = 1000
```

Definition at line 28 of file game.c.

Referenced by gameRoutine(), initValues(), and updateGameLogic().

#### 6.8.3.3 downRoad

```
bool downRoad[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
```

Definition at line 25 of file game.c.

Referenced by initValues(), updateGameDesign(), and updateGameLogic().

#### 6.8.3.4 points

```
int points = 0
```

Definition at line 27 of file game.c.

Referenced by gameRoutine(), initValues(), saveUser(), and updateGameLogic().

#### 6.8.3.5 probabilityToSpawn

```
int probabilityToSpawn = 20
```

Definition at line 26 of file game.c.

Referenced by initValues(), and updateGameLogic().

#### 6.8.3.6 upRoad

```
bool upRoad[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
```

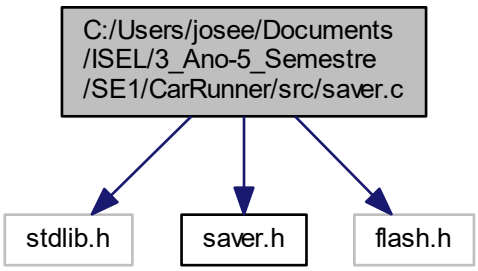
Definition at line 24 of file game.c.

Referenced by initValues(), updateGameDesign(), and updateGameLogic().

## 6.9 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/CarRunner/src/saver.c File Reference

```
#include <stdlib.h>
#include "saver.h"
#include "flash.h"
```

Include dependency graph for saver.c:




### Functions

- void **checkClean** ()
- void **saveScore** (int score, char *name, int name_size)
- int * **readScores** ()

  *reads from flash all the scores*
- int **listSize** ()
- void **readNames** (char *strings[])

  *function that reads from flash all the names its required that strings is started with size[listSize][17] to accommodate for max size names*

### Variables

- int * **names** = **sector28**
- int * **scores** = **sector29**
- int **zero** = 0

### 6.9.1 Function Documentation

#### 6.9.1.1 checkClean()

```
void checkClean ( )
```

Definition at line 15 of file saver.c.

References FLASH_BlanckCheck(), FLASH_WriteData(), IAP_SECTOR_NOT_BLANK, names, scores, and zero.

Referenced by listSize(), readScores(), and saveScore().

#### 6.9.1.2 listSize()

```
int listSize ( )
```

**Returns**

the current number of players that exist

Definition at line 83 of file saver.c.

References checkClean(), and names.

Referenced by playerScoresShowDown().

#### 6.9.1.3 readNames()

```
void readNames (
            char * strings[] )
```

function that reads from flash all the names its required that strings is started with size[listSize][17] to accommodate for max size names

Definition at line 89 of file saver.c.

References names.

Referenced by playerScoresShowDown().

#### 6.9.1.4 readScores()

```
int* readScores ( )
```

reads from flash all the scores

**Returns**

array of scores

Definition at line 70 of file saver.c.

References checkClean(), and scores.

Referenced by gameRoutine(), and playerScoresShowDown().

#### 6.9.1.5 saveScore()

```
void saveScore (
            int score,
            char * name,
            int name_size )
```

Definition at line 22 of file saver.c.

References checkClean(), FLASH_WriteArray(), IAP_CMD_SUCESS, names, and scores.

Referenced by saveUser().

### 6.9.2 Variable Documentation

#### 6.9.2.1 names

```
int* names =  sector28
```

Definition at line 11 of file saver.c.

Referenced by checkClean(), listSize(), readNames(), and saveScore().

#### 6.9.2.2 scores

```
int* scores =  sector29
```

Definition at line 12 of file saver.c.

Referenced by checkClean(), playerScoresShowDown(), readScores(), and saveScore().

#### 6.9.2.3 zero

```
int zero = 0
```

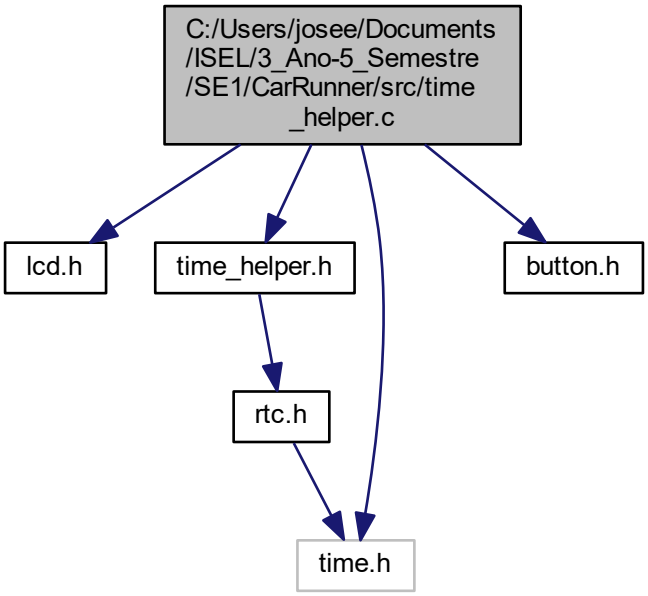
Definition at line 13 of file saver.c.

Referenced by checkClean().

## 6.10 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/CarRunner/src/time_helper.c File Reference

```
#include "lcd.h"
#include "time_helper.h"
#include "button.h"
#include "time.h"
```

Include dependency graph for time_helper.c:




### Functions

- void **Time_changeRoutine** (time_t seconds)

  *routine that interfaces with lcd and buttons to change time in rtc*
- void **update_DateTimeDisplay** (struct tm dateTime)

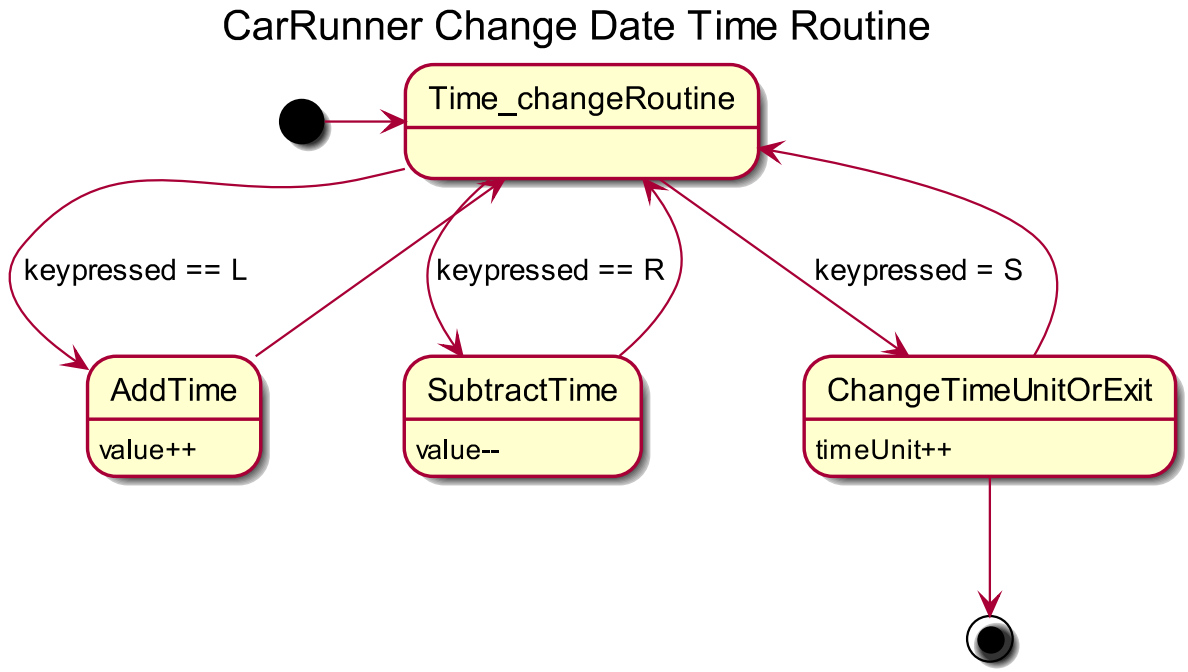
  *updates lcd with date and time*

### 6.10.1 Function Documentation

#### 6.10.1.1 Time_changeRoutine()

```
void Time_changeRoutine (
            time_t seconds )
```

routine that interfaces with lcd and buttons to change time in rtc




Definition at line 30 of file time_helper.c.

References BUTTON_GetButtonsEvents(), key_state::key, LCDText_Clear(), LCDText_CursorOff(), LCDText↩_CursorOn(), LCDText_Locate(), LCDText_WriteString(), PRESSED, RTC_SetValue(), key_state::state, and update_DateTimeDisplay().

Referenced by routineChooser().

#### 6.10.1.2 update_DateTimeDisplay()

```
void update_DateTimeDisplay (
            struct tm datetime )
```

updates lcd with date and time

**Parameters**

| | |
|---|---|
| *datetime* | time to be displayed |

Definition at line 85 of file time_helper.c.

References LCDText_Locate(), and LCDText_WriteString().

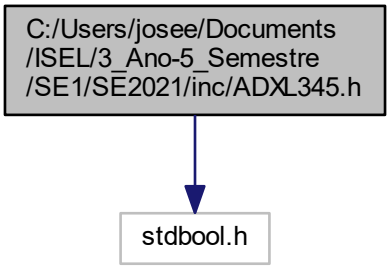
Referenced by main(), and Time_changeRoutine().

## 6.11 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/SE2021/inc/ADXL345.h File Reference

Contains the ADXL345 API.

```
#include <stdbool.h>
```

Include dependency graph for ADXL345.h:




### Macros

- #define **ADXL345_DEVID_RESET_VALUE** (0x00E5)
- #define **ADXL345_REG_DEVID** (0x00)
- #define **ADXL345_REG_THRESH_TAP** (0x1D)
- #define **ADXL345_REG_DUR** (0x21)
- #define **ADXL345_REG_LATENT** (0x22)
- #define **ADXL345_REG_WINDOW** (0x23)
- #define **ADXL345_REG_THRESH_ACT** (0x24)
- #define **ADXL345_REG_TAP_AXES** (0x2A)
- #define **ADXL345_REG_ACT_TAP_STATUS** (0x2B)
- #define **ADXL345_REG_BW_RATE** (0x2C)
- #define **ADXL345_REG_POWER_CTL** (0x2D)
- #define **ADXL345_REG_INT_ENABLE** (0x2E)
- #define **ADXL345_REG_INT_MAP** (0x2F)

- #define **ADXL345_REG_INT_SOURCE** (0x30)
- #define **ADXL345_REG_DATA_FORMAT** (0x31)
- #define **ADXL345_REG_DATAX0** (0x32)
- #define **ADXL345_REG_DATAX1** (0x33)
- #define **ADXL345_REG_DATAY0** (0x34)
- #define **ADXL345_REG_DATAY1** (0x35)
- #define **ADXL345_REG_DATAZ0** (0x36)
- #define **ADXL345_REG_DATAZ1** (0x37)
- #define **ADXL345_GRAVITY_SUN** 273.95f
- #define **ADXL345_GRAVITY_EARTH** 9.80665f
- #define **ADXL345_GRAVITY_MOON** 1.622f
- #define **ADXL345_GRAVITY_MARS** 3.69f
- #define **ADXL345_GRAVITY_NONE** 1.00f

## Enumerations

- enum **ADX345_DataRate_t** {
  **ADXL345_DATARATE_3200HZ** = 0b1111, **ADXL345_DATARATE_1600HZ** = 0b1110, **ADXL345_DAT↩**
  **ARATE_800HZ** = 0b1101, **ADXL345_DATARATE_400HZ** = 0b1100,
  **ADXL345_DATARATE_200HZ** = 0b1011, **ADXL345_DATARATE_100HZ** = 0b1010, **ADXL345_DATA↩**
  **RATE_50HZ** = 0b1001, **ADXL345_DATARATE_25HZ** = 0b1000,
  **ADXL345_DATARATE_12_5HZ** = 0b0111, **ADXL345_DATARATE_6_25HZ** = 0b0110, **ADXL345_DAT↩**
  **ARATE_3_13HZ** = 0b0101, **ADXL345_DATARATE_1_56HZ** = 0b0100,
  **ADXL345_DATARATE_0_78HZ** = 0b0011, **ADXL345_DATARATE_0_39HZ** = 0b0010, **ADXL345_DAT↩**
  **ARATE_0_20HZ** = 0b0001, **ADXL345_DATARATE_0_10HZ** = 0b0000 }
- enum **ADX345_Range_t** { **ADXL345_RANGE_16G** = 0b11, **ADXL345_RANGE_8G** = 0b10, **ADXL345↩**
  **_RANGE_4G** = 0b01, **ADXL345_RANGE_2G** = 0b00 }

## Functions

- int **ADXL345_Init** (void)

  *Initializes the ADXL345 Peripheral.*
- void **setupTap** (float threshold, float duration, float latency, float window)

  *Configs taping.*
- bool **ADXL345_isTap** ()

  *Checks if it was tapped can't be use with ADXL345_isDoubleTap.*
- bool **ADXL345_isDoubleTap** ()

  *Checks if it was double tapped can't be use with ADXL345_isTap.*
- int **ADXL345_Read** (float ∗x_value, float ∗y_value, float ∗z_value)

  *Reads ADXL345 data.*

### 6.11.1 Detailed Description

Contains the ADXL345 API.

**Version**

1.1

**Date**

8 Jan 2020

**Author**

Jose Filipe Cruz dos Santos

## 6.11.2 Enumeration Type Documentation

### 6.11.2.1 ADX345_DataRate_t

enum **ADX345_DataRate_t**

Data Rates

**Enumerator**

| | |
|---|---|
| ADXL345_DATARATE_3200HZ | |
| ADXL345_DATARATE_1600HZ | |
| ADXL345_DATARATE_800HZ | |
| ADXL345_DATARATE_400HZ | |
| ADXL345_DATARATE_200HZ | |
| ADXL345_DATARATE_100HZ | |
| ADXL345_DATARATE_50HZ | |
| ADXL345_DATARATE_25HZ | |
| ADXL345_DATARATE_12_5HZ | |
| ADXL345_DATARATE_6_25HZ | |
| ADXL345_DATARATE_3_13HZ | |
| ADXL345_DATARATE_1_56HZ | |
| ADXL345_DATARATE_0_78HZ | |
| ADXL345_DATARATE_0_39HZ | |
| ADXL345_DATARATE_0_20HZ | |
| ADXL345_DATARATE_0_10HZ | |

Definition at line 95 of file ADXL345.h.

### 6.11.2.2 ADX345_Range_t

enum **ADX345_Range_t**

Data Ranges

**Enumerator**

| | |
|---|---|
| ADXL345_RANGE_16G | |
| ADXL345_RANGE_8G | |
| ADXL345_RANGE_4G | |
| ADXL345_RANGE_2G | |

Definition at line 118 of file ADXL345.h.

### 6.11.3 Function Documentation

#### 6.11.3.1 ADXL345_Init()

```
int ADXL345_Init (
            void  )
```

Initializes the ADXL345 Peripheral.

Definition at line 37 of file ADXL345.c.

References ADXL345_DATARATE_100HZ, ADXL345_DEVID_RESET_VALUE, ADXL345_RANGE_2G, ADX↩L345_REG_BW_RATE, ADXL345_REG_DATA_FORMAT, ADXL345_REG_DEVID, ADXL345_REG_POWER_↩CTL, readRegister(), setupTap(), SPI_ConfigTransfer(), and writeRegister().

Referenced by main().

#### 6.11.3.2 ADXL345_isDoubleTap()

```
bool ADXL345_isDoubleTap ( )
```

Checks if it was double tapped can't be use with ADXL345_isTap.

Definition at line 118 of file ADXL345.c.

References ADXL345_REG_INT_SOURCE, and readRegister().

#### 6.11.3.3 ADXL345_isTap()

```
bool ADXL345_isTap ( )
```

Checks if it was tapped can't be use with ADXL345_isDoubleTap.

Definition at line 112 of file ADXL345.c.

References ADXL345_REG_INT_SOURCE, and readRegister().

Referenced by gameRoutine().

#### 6.11.3.4 ADXL345_Read()

```
int ADXL345_Read (
            float * x_value,
            float * y_value,
            float * z_value )
```

Reads ADXL345 data.

**Parameters**

| | | |
|---|---|---|
| out | *x_value* | x axis value |
| out | *y_value* | y axis value |
| out | *z_value* | z axis value |

X value

Y value

Z value

Definition at line 124 of file ADXL345.c.

References ADXL345_REG_DATAX0, ADXL345_REG_DATAX1, ADXL345_REG_DATAY0, ADXL345_REG_D↩ ATAY1, ADXL345_REG_DATAZ0, ADXL345_REG_DATAZ1, and readRegister().

#### 6.11.3.5 setupTap()

```
void setupTap (
             float threshold,
             float duration,
             float latency,
             float window )
```

Configs taping.

**Parameters**

| | |
|---|---|
| *treshold* | defines treshold |
| *duration* | defines max duration of tap |
| *latency* | defines latency for double tap |
| *window* | defines window for double tap |

Definition at line 78 of file ADXL345.c.

References ADXL345_REG_DUR, ADXL345_REG_INT_ENABLE, ADXL345_REG_INT_MAP, ADXL345_RE↩ G_LATENT, ADXL345_REG_TAP_AXES, ADXL345_REG_THRESH_TAP, ADXL345_REG_WINDOW, read↩ Register(), and writeRegister().

Referenced by ADXL345_Init().

## 6.12 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/S↩ E2021/inc/button.h File Reference

Contains the button API.

## Data Structures

- struct **key_state**

## Macros

- #define **PRESSED** 0
- #define **RELEASED** 1
- #define **REPEATED** 2

## Functions

- void **BUTTON_Init** (void)

  *Initializes the system to premit the access to buttons.*
- int **BUTTON_Hit** (void)

  *Gets pressed button non-blocking version.*
- int **BUTTON_Read** (void)

  *Gets pressed button blocking version.*
- **key_state BUTTON_GetButtonsEvents** (void)

  *Pressed button state.*

### 6.12.1 Detailed Description

Contains the button API.

**Version**

1.1

**Date**

3 Nov 2020

**Author**

Jose Filipe Cruz dos Santos

### 6.12.2 Macro Definition Documentation

#### 6.12.2.1 PRESSED

```
#define PRESSED 0
```

Definition at line 13 of file button.h.

#### 6.12.2.2 RELEASED

```
#define RELEASED 1
```

Definition at line 14 of file button.h.

#### 6.12.2.3 REPEATED

```
#define REPEATED 2
```

Definition at line 15 of file button.h.

### 6.12.3 Function Documentation

#### 6.12.3.1 BUTTON_GetButtonsEvents()

```
key_state BUTTON_GetButtonsEvents (
            void  )
```

Pressed button state.

**Returns**

button state code (bitmap): pressed ,released , repeated

Definition at line 72 of file button.c.

References BUTTON_Hit(), and keyState.

Referenced by gameRoutine(), main(), Menu(), playerScoresShowDown(), saveUser(), and Time_changeRoutine().

#### 6.12.3.2 BUTTON_Hit()

```
int BUTTON_Hit (
            void  )
```

Gets pressed button non-blocking version.

**Returns**

button code (bitmap) if pressed otherwise -1

Definition at line 41 of file button.c.

References BUTTON_ONE, key_state::key, keyState, PRESSED, RELEASED, REPEATED, key_state::state, and WAIT_ChronoUs().

Referenced by BUTTON_GetButtonsEvents(), BUTTON_Init(), and BUTTON_Read().

#### 6.12.3.3 BUTTON_Init()

```
void BUTTON_Init (
            void  )
```

Initializes the system to premit the access to buttons.

Initializes the system to premit the access to buttons.

Definition at line 32 of file button.c.

References BUTTON_Hit(), BUTTON_ONE, and keyState.

Referenced by main().

#### 6.12.3.4 BUTTON_Read()

```
int BUTTON_Read (
            void  )
```

Gets pressed button blocking version.

**Returns**

button code (bitmap) when pressed

Gets pressed button blocking version.

Definition at line 64 of file button.c.

References BUTTON_Hit().

## 6.13 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/SE2021/inc/Flash.h File Reference

### Macros

- #define **IAP_CMD_SUCESS** 0
- #define **IAP_INVALID_COMMAND** 1
- #define **IAP_SRC_ADDR_ERROR** 2
- #define **IAP_DST_ADDR_ERROR** 3
- #define **IAP_SRC_ADDR_NOT_MAPPED** 4
- #define **IAP_DST_ADDR_NOT_MAPPED** 5
- #define **IAP_COUNT_ERROR** 6
- #define **IAP_INVALID_SECTOR** 7
- #define **IAP_SECTOR_NOT_BLANK** 8
- #define **IAP_SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION** 9
- #define **IAP_COMPARE_ERROR** 10
- #define **IAP_BUSY** 11
- #define **SECTOR_SIZE** 32768
- #define **sector28** 0x00070000
- #define **sector29** 0x00078000

## Functions

- unsigned int **FLASH_EraseSectors** (unsigned int startSector, unsigned int endSector)

  *Deletes the contents of a sector, or multiple sectors, of FLASH. To delete only one sector, the same sector number must be used for both parameters.*
- unsigned int **FLASH_WriteData** (void ∗dstAddr, void ∗srcAddr, unsigned int size)

  *Write the data block. This Function can only access sectors from sector 16 to 29.*
- unsigned int **FLASH_VerifyData** (void ∗dstAddr, void ∗srcAddr, unsigned int size)

  *Compares the contents of the data block.*
- unsigned int **FLASH_WriteArray** (void ∗dstAddr, int srcAddr[ ], int array_size, unsigned int size)

  *Writes an array to a data block. This Function can only access sectors from sector 16 to 29.*
- int **FLASH_WriteBlock** (void ∗dstAddr, void ∗srcAddr, unsigned int size)
- unsigned int **FLASH_BlanckCheck** (unsigned int startSector, unsigned int endSector)

### 6.13.1 Macro Definition Documentation

#### 6.13.1.1 sector28

```
#define sector28 0x00070000
```

Sector 28 adrress

Definition at line 36 of file Flash.h.

#### 6.13.1.2 sector29

```
#define sector29 0x00078000
```

Sector 29 adrress

Definition at line 39 of file Flash.h.

#### 6.13.1.3 SECTOR_SIZE

```
#define SECTOR_SIZE 32768
```

Sectors sizes

Definition at line 33 of file Flash.h.

### 6.13.2 Function Documentation

#### 6.13.2.1 FLASH_BlanckCheck()

```
unsigned int FLASH_BlanckCheck (
             unsigned int startSector,
             unsigned int endSector )
```

Definition at line 115 of file Flash.c.

References iap_entry.

Referenced by checkClean().

#### 6.13.2.2 FLASH_EraseSectors()

```
unsigned int FLASH_EraseSectors (
             unsigned int startSector,
             unsigned int endSector )
```

Deletes the contents of a sector, or multiple sectors, of FLASH. To delete only one sector, the same sector number must be used for both parameters.

Definition at line 16 of file Flash.c.

References IAP_CMD_SUCESS, and iap_entry.

Referenced by FLASH_WriteArray(), and FLASH_WriteData().

#### 6.13.2.3 FLASH_VerifyData()

```
unsigned int FLASH_VerifyData (
             void * dstAddr,
             void * srcAddr,
             unsigned int size )
```

Compares the contents of the data block.

**Parameters**

| | |
|---|---|
| *srcAddr* | data block reference |
| *size* | size in bytes of the data block |
| *dstAddr* | data block to be compared to |

Definition at line 70 of file Flash.c.

References iap_entry.

#### 6.13.2.4 FLASH_WriteArray()

```
unsigned int FLASH_WriteArray (
             void * dstAddr,
             int srcAddr[],
             int array_size,
             unsigned int size )
```

Writes an array to a data block. This Function can only access sectors from sector 16 to 29.

**Parameters**

| | |
|---|---|
| *srcAddr* | array to be written |
| *array_size* | size of the array |
| *size* | block size |
| *dstAddr* | Flash address |

Definition at line 83 of file Flash.c.

References FLASH_EraseSectors(), IAP_CMD_SUCESS, iap_entry, and SECTOR_SIZE.

Referenced by saveScore().

#### 6.13.2.5 FLASH_WriteBlock()

```
int FLASH_WriteBlock (
             void * dstAddr,
             void * srcAddr,
             unsigned int size )
```

Definition at line 53 of file Flash.c.

References iap_entry.

Referenced by FLASH_WriteData().

#### 6.13.2.6 FLASH_WriteData()

```
unsigned int FLASH_WriteData (
             void * dstAddr,
             void * srcAddr,
             unsigned int size )
```

Write the data block. This Function can only access sectors from sector 16 to 29.

**Parameters**

| | |
|---|---|
| *srcAddr* | data block pointer |
| *size* | block size |
| *dstAddr* | Flash address |

Definition at line 36 of file Flash.c.

References FLASH_EraseSectors(), FLASH_WriteBlock(), IAP_CMD_SUCESS, iap_entry, and SECTOR_SIZE.

Referenced by checkClean().

## 6.14 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/SE2021/inc/lcd.h File Reference

Contains the flash API.

### Macros

- #define **LCDText_LINES** 2
- #define **LCDText_COLUMNS** 16
- #define **LCDText_LINE_OFFSET** 0x40
- #define **LCDText_CMD** 0
- #define **LCDText_DATA** 1
- #define **LCDText_CMD_DISPLAY_OFF** 0x08
- #define **LCDText_CMD_DISPLAY_ON** 0x0C
- #define **LCDText_CMD_DISPLAY_CLEAR** 1
- #define **LCDText_CMD_ENTRY_MODE_SET** 0x06
- #define **LCDText_CMD_FUNCTION_SET** 0x28
- #define **LCDText_CMD_RETURN_HOME** 2
- #define **LCDText_CMD_SET_DDRAM_ADDR** 0x80

### Functions

- void **LCDText_WriteCmd** (char cmd)

  *Gives an instruction to the LCD controller.*
- void **LCDText_WriteChar** (char ch)

  *Writes a character at the current cursor position.*
- void **LCDText_Init** (void)

  *Initiates the system to allow access to the LCD peripheral, using 2 lines with 16 columns and 4-bit communication.*
- void **LCDText_WriteString** (char ∗str)

  *Writes a string at the current cursor position.*
- void **LCDText_Locate** (int row, int column)

  *Positions the cursor on the row and column of the display.*
- void **LCDText_Clear** (void)

  *Clear the display using the command available in the peripheral API.*
- void **LCDText_CreateChar** (unsigned char location, unsigned char charmap[ ])

  *Creates a character.*
- void **LCDText_Printf** (char ∗fmt,...)

  *Writes the string fmt at the current cursor position.*
- void **LCDText_CursorOn** ()

  *Turns on display cursor on.*
- void **LCDText_CursorOff** ()

  *Turns on display cursor off.*

### 6.14.1 Detailed Description

Contains the flash API.

Contains the lcd API.

**Version**

1.1

**Date**

12 Jan 2020

**Author**

Jose Filipe Cruz dos Santos

**Version**

1.1

**Date**

17 Nov 2020

**Author**

Jose Filipe Cruz dos Santos

### 6.14.2 Macro Definition Documentation

#### 6.14.2.1 LCDText_CMD

```
#define LCDText_CMD 0
```

Definition at line 27 of file lcd.h.

#### 6.14.2.2 LCDText_DATA

```
#define LCDText_DATA 1
```

Definition at line 28 of file lcd.h.

#### 6.14.2.3 LCDText_LINE_OFFSET

```
#define LCDText_LINE_OFFSET 0x40
```

DISPLAY_DDRAM_ADDRESSING_OFFSET

Definition at line 25 of file lcd.h.

### 6.14.3 Function Documentation

#### 6.14.3.1 LCDText_Clear()

```
void LCDText_Clear (
            void  )
```

Clear the display using the command available in the peripheral API.

Definition at line 129 of file lcd.c.

References LCDText_CMD_DISPLAY_CLEAR, LCDText_Locate(), LCDText_WriteCmd(), and WAIT_ChronoUs().

Referenced by gameRoutine(), LCDText_Init(), Menu(), playerScoresShowDown(), saveUser(), and Time_change↩ Routine().

#### 6.14.3.2 LCDText_CreateChar()

```
void LCDText_CreateChar (
            unsigned char location,
            unsigned char charmap[] )
```

Creates a character.

**Parameters**

| | |
|---|---|
| *location* | position in the character table |
| *charmap* | character design |

Definition at line 69 of file lcd.c.

References LCDText_WriteByte(), LCDText_WriteCmd(), and WAIT_ChronoUs().

Referenced by gameInit(), and LCDText_Init().

#### 6.14.3.3 LCDText_CursorOff()

```
void LCDText_CursorOff ( )
```

Turns on display cursor off.

Definition at line 148 of file lcd.c.

References LCDText_CMD_DISPLAY_ON, and LCDText_WriteCmd().

Referenced by saveUser(), and Time_changeRoutine().

#### 6.14.3.4 LCDText_CursorOn()

```
void LCDText_CursorOn ( )
```

Turns on display cursor on.

Definition at line 144 of file lcd.c.

References LCDText_CMD_DISPLAY_ON, and LCDText_WriteCmd().

Referenced by saveUser(), and Time_changeRoutine().

#### 6.14.3.5 LCDText_Init()

```
void LCDText_Init (
            void  )
```

Initiates the system to allow access to the LCD peripheral, using 2 lines with 16 columns and 4-bit communication.

Definition at line 45 of file lcd.c.

References DB4, EN, LCDText_Clear(), LCDText_CMD_DISPLAY_OFF, LCDText_CMD_DISPLAY_ON, LCDText↩_CMD_ENTRY_MODE_SET, LCDText_CMD_FUNCTION_SET, LCDText_CreateChar(), LCDText_Locate(), LC↩DText_WriteCmd(), LCDText_WriteNibble(), RS, and WAIT_ChronoUs().

Referenced by main().

#### 6.14.3.6 LCDText_Locate()

```
void LCDText_Locate (
            int row,
            int column )
```

Positions the cursor on the row and column of the display.

Coloca o cursor na posicao (x,y)

**Parameters**

| | |
|---|---|
| *row* | from 1 to 2 |
| *column* | from 0 to 15 |

Definition at line 114 of file lcd.c.

References LCDText_CMD_SET_DDRAM_ADDR, LCDText_COLUMNS, LCDText_LINE_OFFSET, LCDText_LI↩ NES, LCDText_WriteCmd(), x, and y.

Referenced by gameRoutine(), LCDText_Clear(), LCDText_Init(), LCDText_WriteString(), saveUser(), Time_↩ changeRoutine(), update_DateTimeDisplay(), and updateGameDesign().

#### 6.14.3.7 LCDText_Printf()

```
void LCDText_Printf (
             char * fmt,
              ...  )
```

Writes the string fmt at the current cursor position.

**Parameters**

| | |
|---|---|
| *fmt* | format of the string |

Definition at line 135 of file lcd.c.

References LCDText_WriteString().

Referenced by gameRoutine(), and playerScoresShowDown().

#### 6.14.3.8 LCDText_WriteChar()

```
void LCDText_WriteChar (
             char ch )
```

Writes a character at the current cursor position.

Definition at line 86 of file lcd.c.

References LCDText_DATA, LCDText_WriteByte(), WAIT_ChronoUs(), and x.

Referenced by LCDText_WriteString(), and saveUser().

#### 6.14.3.9 LCDText_WriteCmd()

```
void LCDText_WriteCmd (
            char cmd )
```

Gives an instruction to the LCD controller.

Definition at line 81 of file lcd.c.

References LCDText_CMD, LCDText_WriteByte(), and WAIT_ChronoUs().

Referenced by LCDText_Clear(), LCDText_CreateChar(), LCDText_CursorOff(), LCDText_CursorOn(), LCDText_↩ Init(), and LCDText_Locate().

#### 6.14.3.10 LCDText_WriteString()

```
void LCDText_WriteString (
            char * str )
```

Writes a string at the current cursor position.

Definition at line 92 of file lcd.c.

References LCDText_COLUMNS, LCDText_LINES, LCDText_Locate(), LCDText_WriteChar(), WAIT_ChronoUs(), x, and y.

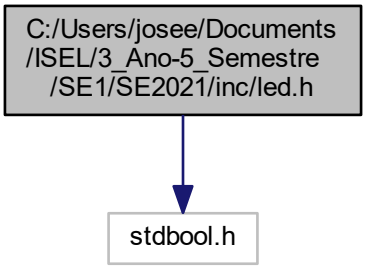
Referenced by gameRoutine(), LCDText_Printf(), Menu(), playerScoresShowDown(), saveUser(), Time_change↩ Routine(), update_DateTimeDisplay(), and updateGameDesign().

## 6.15 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/S↩ E2021/inc/led.h File Reference

Contains the led peripheral manager API.

```
#include <stdbool.h>
```

Include dependency graph for led.h:



## Functions

- void **LED_Init** (bool state)

  *Initiates the system to allow manipulation of the status LED that exists on the LPCXpresso LPC1769 prototyping board.*
- bool **LED_GetState** (void)
- void **LED_On** (void)

  *Turns on the LED.*
- void **LED_Off** (void)

  *Turns off the LED.*

### 6.15.1 Detailed Description

Contains the led peripheral manager API.

**Version**

1.1

**Date**

3 Nov 2020

**Author**

Jose Filipe Cruz dos Santos

### 6.15.2 Function Documentation

#### 6.15.2.1 LED_GetState()

```
bool LED_GetState (
            void  )
```

**Returns**

Returns true if the LED is on and false if the LED is off.

Definition at line 26 of file led.c.

References led_state.

#### 6.15.2.2 LED_Init()

```
void LED_Init (
            bool state )
```

Initiates the system to allow manipulation of the status LED that exists on the LPCXpresso LPC1769 prototyping board.

**Parameters**

| | |
|---|---|
| *state* | Leave the LED off when the status is false or lit when true. |

Definition at line 16 of file led.c.

References led_state.

#### 6.15.2.3 LED_Off()

```
void LED_Off (
              void  )
```

Turns off the LED.

Definition at line 35 of file led.c.

References led_state.

#### 6.15.2.4 LED_On()

```
void LED_On (
              void  )
```

Turns on the LED.

Definition at line 30 of file led.c.

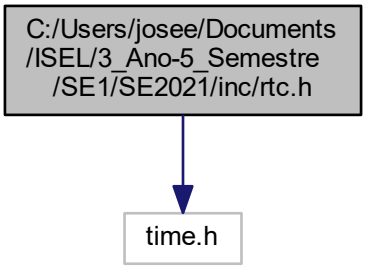
References led_state.

## 6.16 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/SE2021/inc/rtc.h File Reference

Contains the rtc peripheral manager API.

```
#include "time.h"
```

Include dependency graph for rtc.h:



## Functions

- void **RTC_Init** (time_t seconds)

  *Initiates the system to allow access to the RTC peripheral.*
- void **RTC_GetValue** (struct tm ∗dateTime)
- void **RTC_SetValue** (struct tm ∗dateTime)

  *Updates RTC.*
- void **RTC_SetSeconds** (time_t seconds)

  *Performs the RTC update with the value of the seconds parameter.*
- time_t **RTC_GetSeconds** (void)

### 6.16.1 Detailed Description

Contains the rtc peripheral manager API.

**Version**

1.1

**Date**

3 Nov 2020

**Author**

Jose Filipe Cruz dos Santos

### 6.16.2 Function Documentation

#### 6.16.2.1 RTC_GetSeconds()

```
time_t RTC_GetSeconds (
            void  )
```

**Returns**

Returns the current value of the RTC, in seconds since 00:00:00 UTC on January 1 1970

Definition at line 64 of file rtc.c.

References RTC_GetValue().

Referenced by routineChooser().

#### 6.16.2.2 RTC_GetValue()

```
void RTC_GetValue (
            struct tm * dateTime )
```

**Parameters**

| | | |
|---|---|---|
| out | *dateTime* | current RTC value ajusted to 00:00:00 UTC from 1 January 1970. |

Definition at line 25 of file rtc.c.

Referenced by main(), and RTC_GetSeconds().

#### 6.16.2.3 RTC_Init()

```
void RTC_Init (
            time_t seconds )
```

Initiates the system to allow access to the RTC peripheral.

**Parameters**

| | |
|---|---|
| *seconds* | The RTC is started with this value, which represents the seconds since 00:00:00 UTC from 1 January 1970. |

Definition at line 14 of file rtc.c.

References RTC_SetSeconds().

Referenced by main().

#### 6.16.2.4 RTC_SetSeconds()

```
void RTC_SetSeconds (
            time_t seconds )
```

Performs the RTC update with the value of the seconds parameter.

**Parameters**

| | |
|---|---|
| *seconds* | represents the seconds since 00:00:00 UTC of 1 January 1970. |

Definition at line 59 of file rtc.c.

References RTC_SetValue().

Referenced by RTC_Init().

#### 6.16.2.5 RTC_SetValue()

```
void RTC_SetValue (
            struct tm * dateTime )
```

Updates RTC.

**Parameters**

| | |
|---|---|
| *dateTime* | value that is set to update RTC. |

Definition at line 45 of file rtc.c.

Referenced by RTC_SetSeconds(), and Time_changeRoutine().

## 6.17 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/SE2021/inc/spi.h File Reference

Contains the spi peripheral manager API.

### Functions

- void **SPI_Init** (void)
  *Faz a iniciação do controlador e configura os respetivos pinos.*
- void **SPI_ConfigTransfer** (int frequency, int bitData, int mode)
  *Configures the transfer.*
- int **SPI_Transfer** (unsigned short *txBuffer, unsigned short *rxBuffer, int lenght)
  *Makes a transfer. Returns the success or error in the transfer.*

### 6.17.1 Detailed Description

Contains the spi peripheral manager API.

**Version**

1.1

**Date**

17 Dec 2020

**Author**

Jose Filipe Cruz dos Santos

## 6.17.2 Function Documentation

### 6.17.2.1 SPI_ConfigTransfer()

```
void SPI_ConfigTransfer (
            int frequency,
            int bitData,
            int mode )
```

Configures the transfer.

**Parameters**

| | |
|---|---|
| *frequency* | send/reception frequency |
| *bitData* | number of bits of changed data |
| *mode* | transfer mode is a two bit value (1st bit = CPOL, 2nd bit = CPHA). |

Definition at line 15 of file spi.c.

Referenced by ADXL345_Init().

### 6.17.2.2 SPI_Init()

```
void SPI_Init (
            void  )
```

Faz a iniciação do controlador e configura os respetivos pinos.

Definition at line 7 of file spi.c.

Referenced by main().

### 6.17.2.3 SPI_Transfer()

```
int SPI_Transfer (
            unsigned short * txBuffer,
            unsigned short * rxBuffer,
            int lenght )
```

Makes a transfer. Returns the success or error in the transfer.

**Parameters**

| | |
|---|---|
| *txBuffer* | transfer ruffer contains transfer data |
| *rxBuffer* | recieve buffer contains recieved data |
| *length* | how many data is there to be transfered/recieved |

Definition at line 35 of file spi.c.

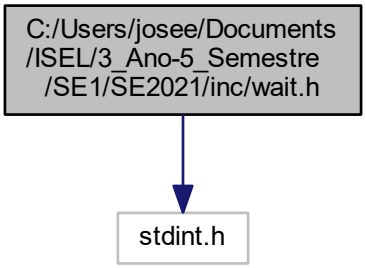
Referenced by readRegister(), and writeRegister().

## 6.18 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/SE2021/inc/wait.h File Reference

Contains the delay API.

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

Include dependency graph for wait.h:




### Functions

- int32_t **WAIT_Init** (void)

  *Initialises the wait API for 1 ms resolution.*
- void **WAIT_Milliseconds** (uint32_t millis)

  *Waits a number of milliseconds.*
- uint32_t **WAIT_GetElapsedMillis** (uint32_t start)

  *Get difference in milliseconds from parameter.*
- void **WAIT_ChronoUs** (uint32_t waitUs)

  *Waits waitUs microseconds.*

### 6.18.1 Detailed Description

Contains the delay API.

**Version**

1.1

**Date**

10 Out 2017

**Author**

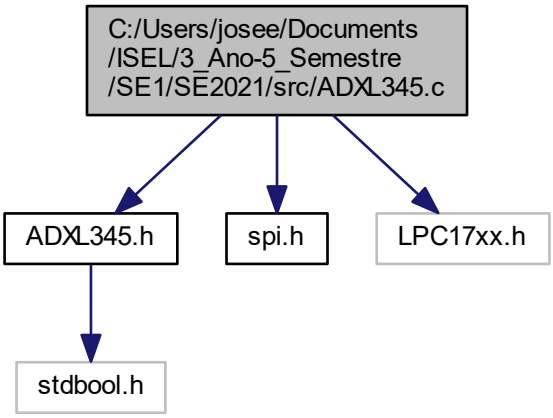
PSampaio



## 6.19 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/SE2021/src/ADXL345.c File Reference

```
#include "ADXL345.h"
#include "spi.h"
#include "LPC17xx.h"
```

Include dependency graph for ADXL345.c:




### Macros

- #define **constrain**(amt, low, high) ((amt)<(low)?(low):((amt)>(high)?(high):(amt)))

## Functions

- void **writeRegister** (unsigned short reg, unsigned short value)
- void **readRegisters** (unsigned short ∗reg, unsigned short value, int size)
- void **readRegister** (unsigned short reg, unsigned short ∗retValue)
- int **ADXL345_Init** ()

  *Initializes the ADXL345 Peripheral.*
- void **setupTap** (float threshold, float duration, float latency, float window)

  *Configs taping.*
- bool **ADXL345_isTap** ()

  *Checks if it was tapped can't be use with ADXL345_isDoubleTap.*
- bool **ADXL345_isDoubleTap** ()

  *Checks if it was double tapped can't be use with ADXL345_isTap.*
- int **ADXL345_Read** (float ∗x_value, float ∗y_value, float ∗z_value)

  *Reads ADXL345 data.*

### 6.19.1 Macro Definition Documentation

#### 6.19.1.1 constrain

```
#define constrain(
              amt,
              low,
              high ) ((amt)<(low)?(low):((amt)>(high)?(high):(amt)))
```

Definition at line 5 of file ADXL345.c.

### 6.19.2 Function Documentation

#### 6.19.2.1 ADXL345_Init()

```
int ADXL345_Init (
             void  )
```

Initializes the ADXL345 Peripheral.

Definition at line 37 of file ADXL345.c.

References ADXL345_DATARATE_100HZ, ADXL345_DEVID_RESET_VALUE, ADXL345_RANGE_2G, ADX↩ L345_REG_BW_RATE, ADXL345_REG_DATA_FORMAT, ADXL345_REG_DEVID, ADXL345_REG_POWER_↩ CTL, readRegister(), setupTap(), SPI_ConfigTransfer(), and writeRegister().

Referenced by main().

#### 6.19.2.2 ADXL345_isDoubleTap()

```
bool ADXL345_isDoubleTap ( )
```

Checks if it was double tapped can't be use with ADXL345_isTap.

Definition at line 118 of file ADXL345.c.

References ADXL345_REG_INT_SOURCE, and readRegister().

#### 6.19.2.3 ADXL345_isTap()

```
bool ADXL345_isTap ( )
```

Checks if it was tapped can't be use with ADXL345_isDoubleTap.

Definition at line 112 of file ADXL345.c.

References ADXL345_REG_INT_SOURCE, and readRegister().

Referenced by gameRoutine().

#### 6.19.2.4 ADXL345_Read()

```
int ADXL345_Read (
            float * x_value,
            float * y_value,
            float * z_value )
```

Reads ADXL345 data.

**Parameters**

| | | |
|---|---|---|
| out | *x_value* | x axis value |
| out | *y_value* | y axis value |
| out | *z_value* | z axis value |

X value

Y value

Z value

Definition at line 124 of file ADXL345.c.

References ADXL345_REG_DATAX0, ADXL345_REG_DATAX1, ADXL345_REG_DATAY0, ADXL345_REG_D↩ ATAY1, ADXL345_REG_DATAZ0, ADXL345_REG_DATAZ1, and readRegister().

#### 6.19.2.5 readRegister()

```
void readRegister (
            unsigned short reg,
            unsigned short * retValue )
```

Definition at line 23 of file ADXL345.c.

References SPI_Transfer().

Referenced by ADXL345_Init(), ADXL345_isDoubleTap(), ADXL345_isTap(), ADXL345_Read(), and setupTap().

#### 6.19.2.6 readRegisters()

```
void readRegisters (
            unsigned short * reg,
            unsigned short value,
            int size )
```

Definition at line 19 of file ADXL345.c.

#### 6.19.2.7 setupTap()

```
void setupTap (
            float threshold,
            float duration,
            float latency,
            float window )
```

Configs taping.

**Parameters**

| | |
|---|---|
| *treshold* | defines treshold |
| *duration* | defines max duration of tap |
| *latency* | defines latency for double tap |
| *window* | defines window for double tap |

Definition at line 78 of file ADXL345.c.

References ADXL345_REG_DUR, ADXL345_REG_INT_ENABLE, ADXL345_REG_INT_MAP, ADXL345_RE↩G_LATENT, ADXL345_REG_TAP_AXES, ADXL345_REG_THRESH_TAP, ADXL345_REG_WINDOW, read↩Register(), and writeRegister().

Referenced by ADXL345_Init().

#### 6.19.2.8 writeRegister()

```
void writeRegister (
            unsigned short reg,
            unsigned short value )
```

Definition at line 7 of file ADXL345.c.

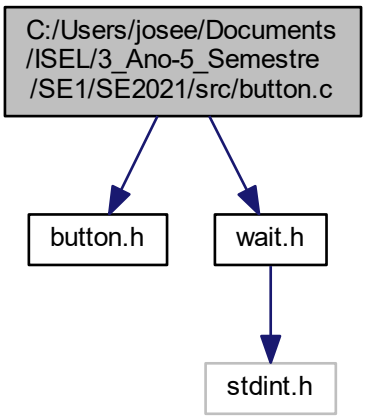
References SPI_Transfer().

Referenced by ADXL345_Init(), and setupTap().

## 6.20 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/SE2021/src/button.c File Reference

```
#include "button.h"
#include "wait.h"
```

Include dependency graph for button.c:




### Macros

- #define **BUTTON_ONE** 24
- #define **BUTTON_TWO** 25
- #define **BUTTON_THREE** 26

## Functions

- void **BUTTON_Init** (void)

  *Initializes every button and sets mode for to pull-up.*
- int **BUTTON_Hit** (void)

  *Gets pressed button non-blocking version.*
- int **BUTTON_Read** (void)

  *Blocks while button_Hit result equals -1.*
- **key_state BUTTON_GetButtonsEvents** ()

  *Pressed button state.*

## Variables

- **key_state keyState**

## 6.20.1 Function Documentation

### 6.20.1.1 BUTTON_GetButtonsEvents()

```
key_state BUTTON_GetButtonsEvents (
            void  )
```

Pressed button state.

**Returns**

button state code (bitmap): pressed ,released , repeated

Definition at line 72 of file button.c.

References BUTTON_Hit(), and keyState.

Referenced by gameRoutine(), main(), Menu(), playerScoresShowDown(), saveUser(), and Time_changeRoutine().

### 6.20.1.2 BUTTON_Hit()

```
int BUTTON_Hit (
            void  )
```

Gets pressed button non-blocking version.

**Returns**

button code (bitmap) if pressed otherwise -1

Definition at line 41 of file button.c.

References BUTTON_ONE, key_state::key, keyState, PRESSED, RELEASED, REPEATED, key_state::state, and WAIT_ChronoUs().

Referenced by BUTTON_GetButtonsEvents(), BUTTON_Init(), and BUTTON_Read().

#### 6.20.1.3 BUTTON_Init()

```
void BUTTON_Init (
             void  )
```

Initializes every button and sets mode for to pull-up.

Initializes the system to premit the access to buttons.

Definition at line 32 of file button.c.

References BUTTON_Hit(), BUTTON_ONE, and keyState.

Referenced by main().

#### 6.20.1.4 BUTTON_Read()

```
int BUTTON_Read (
             void  )
```

Blocks while button_Hit result equals -1.

Gets pressed button blocking version.

Definition at line 64 of file button.c.

References BUTTON_Hit().

### 6.20.2 Variable Documentation

#### 6.20.2.1 keyState

```
key_state keyState
```

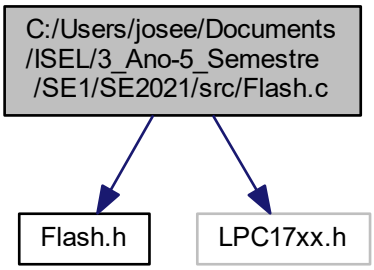
Definition at line 27 of file button.c.

Referenced by BUTTON_GetButtonsEvents(), BUTTON_Hit(), BUTTON_Init(), main(), Menu(), and playerScores↩ ShowDown().

# 6.21 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/SE2021/src/Flash.c File Reference

```
#include "Flash.h"
#include "LPC17xx.h"
```

Include dependency graph for Flash.c:




## Macros

- #define **IAP_LOCATION** 0x1FFF1FF1

## Typedefs

- typedef void(∗ **IAP**) (unsigned int[ ], unsigned int[ ])

## Functions

- unsigned int **FLASH_EraseSectors** (unsigned int startSector, unsigned int endSector)

  *Deletes the contents of a sector, or multiple sectors, of FLASH. To delete only one sector, the same sector number must be used for both parameters.*
- unsigned int **FLASH_WriteData** (void ∗dstAddr, void ∗srcAddr, unsigned int size)

  *Write the data block. This Function can only access sectors from sector 16 to 29.*
- int **FLASH_WriteBlock** (void ∗dstAddr, void ∗srcAddr, unsigned int size)
- unsigned int **FLASH_VerifyData** (void ∗dstAddr, void ∗srcAddr, unsigned int size)

  *Compares the contents of the data block.*
- unsigned int **FLASH_WriteArray** (void ∗dstAddr, int srcAddr[ ], int array_size, unsigned int size)

  *Writes an array to a data block. This Function can only access sectors from sector 16 to 29.*
- unsigned int **FLASH_BlanckCheck** (unsigned int startSector, unsigned int endSector)

## Variables

- **IAP iap_entry** = ( **IAP**) **IAP_LOCATION**

### 6.21.1 Macro Definition Documentation

#### 6.21.1.1 IAP_LOCATION

```
#define IAP_LOCATION 0x1FFF1FF1
```

Definition at line 12 of file Flash.c.

### 6.21.2 Typedef Documentation

#### 6.21.2.1 IAP

```
typedef void(* IAP) (unsigned int[], unsigned int[])
```

Definition at line 13 of file Flash.c.

### 6.21.3 Function Documentation

#### 6.21.3.1 FLASH_BlanckCheck()

```
unsigned int FLASH_BlanckCheck (
            unsigned int startSector,
            unsigned int endSector )
```

Definition at line 115 of file Flash.c.

References iap_entry.

Referenced by checkClean().

#### 6.21.3.2 FLASH_EraseSectors()

```
unsigned int FLASH_EraseSectors (
            unsigned int startSector,
            unsigned int endSector )
```

Deletes the contents of a sector, or multiple sectors, of FLASH. To delete only one sector, the same sector number must be used for both parameters.

Definition at line 16 of file Flash.c.

References IAP_CMD_SUCESS, and iap_entry.

Referenced by FLASH_WriteArray(), and FLASH_WriteData().

#### 6.21.3.3 FLASH_VerifyData()

```
unsigned int FLASH_VerifyData (
            void * dstAddr,
            void * srcAddr,
            unsigned int size )
```

Compares the contents of the data block.

**Parameters**

| | |
|---|---|
| *srcAddr* | data block reference |
| *size* | size in bytes of the data block |
| *dstAddr* | data block to be compared to |

Definition at line 70 of file Flash.c.

References iap_entry.

#### 6.21.3.4 FLASH_WriteArray()

```
unsigned int FLASH_WriteArray (
            void * dstAddr,
            int srcAddr[],
            int array_size,
            unsigned int size )
```

Writes an array to a data block. This Function can only access sectors from sector 16 to 29.

**Parameters**

| | |
|---|---|
| *srcAddr* | array to be written |
| *array_size* | size of the array |
| *size* | block size |
| *dstAddr* | Flash address |

Definition at line 83 of file Flash.c.

References FLASH_EraseSectors(), IAP_CMD_SUCESS, iap_entry, and SECTOR_SIZE.

Referenced by saveScore().

#### 6.21.3.5 FLASH_WriteBlock()

```
int FLASH_WriteBlock (
            void * dstAddr,
```

```
            void * srcAddr,
            unsigned int size )
```

Definition at line 53 of file Flash.c.

References iap_entry.

Referenced by FLASH_WriteData().

#### 6.21.3.6 FLASH_WriteData()

```
unsigned int FLASH_WriteData (
            void * dstAddr,
            void * srcAddr,
            unsigned int size )
```

Write the data block. This Function can only access sectors from sector 16 to 29.

**Parameters**

| | |
|---|---|
| *srcAddr* | data block pointer |
| *size* | block size |
| *dstAddr* | Flash address |

Definition at line 36 of file Flash.c.

References FLASH_EraseSectors(), FLASH_WriteBlock(), IAP_CMD_SUCESS, iap_entry, and SECTOR_SIZE.

Referenced by checkClean().

### 6.21.4 Variable Documentation

#### 6.21.4.1 iap_entry

```
IAP iap_entry = ( IAP)  IAP_LOCATION
```

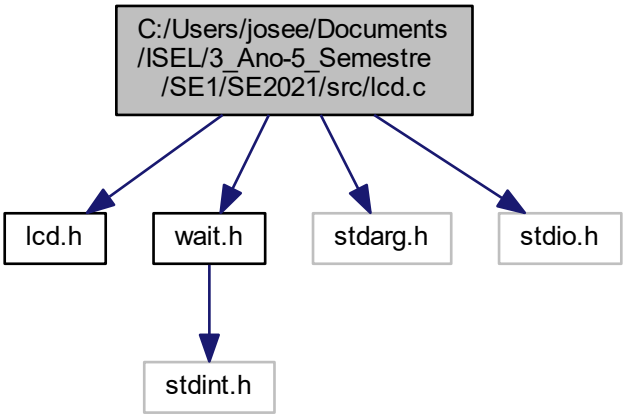
Definition at line 14 of file Flash.c.

Referenced by FLASH_BlanckCheck(), FLASH_EraseSectors(), FLASH_VerifyData(), FLASH_WriteArray(), FLA↩SH_WriteBlock(), and FLASH_WriteData().

# 6.22 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/SE2021/src/lcd.c File Reference

```
#include "lcd.h"
#include "wait.h"
#include <stdarg.h>
#include <stdio.h>
```

Include dependency graph for lcd.c:




## Macros

- #define **EN** 0
- #define **RS** 1
- #define **DB4** 6

## Functions

- void **LCDText_WriteNibble** (char data, unsigned int rs)
- void **LCDText_WriteByte** (char data, unsigned int rs)
- void **LCDText_Init** (void)

  *Initiates the system to allow access to the LCD peripheral, using 2 lines with 16 columns and 4-bit communication.*
- void **LCDText_CreateChar** (unsigned char location, unsigned char charmap[ ])

  *Creates a character.*
- void **LCDText_WriteCmd** (char cmd)

  *Gives an instruction to the LCD controller.*
- void **LCDText_WriteChar** (char ch)

  *Writes a character at the current cursor position.*
- void **LCDText_WriteString** (char ∗str)

  *Writes a string at the current cursor position.*

- void **LCDText_Locate** (int row, int column)

  *Positions the cursor on the row and column of the display.*
- void **LCDText_Clear** (void)

  *Clear the display using the command available in the peripheral API.*
- void **LCDText_Printf** (char *fmt,...)

  *Writes the string fmt at the current cursor position.*
- void **LCDText_CursorOn** ()

  *Turns on display cursor on.*
- void **LCDText_CursorOff** ()

  *Turns on display cursor off.*

## Variables

- int **x**
- int **y**

### 6.22.1 Macro Definition Documentation

#### 6.22.1.1 DB4

```
#define DB4 6
```

Definition at line 19 of file lcd.c.

#### 6.22.1.2 EN

```
#define EN 0
```

Definition at line 17 of file lcd.c.

#### 6.22.1.3 RS

```
#define RS 1
```

Definition at line 18 of file lcd.c.

### 6.22.2 Function Documentation

#### 6.22.2.1 LCDText_Clear()

```
void LCDText_Clear (
            void  )
```

Clear the display using the command available in the peripheral API.

Definition at line 129 of file lcd.c.

References LCDText_CMD_DISPLAY_CLEAR, LCDText_Locate(), LCDText_WriteCmd(), and WAIT_ChronoUs().

Referenced by gameRoutine(), LCDText_Init(), Menu(), playerScoresShowDown(), saveUser(), and Time_change↩ Routine().

#### 6.22.2.2 LCDText_CreateChar()

```
void LCDText_CreateChar (
            unsigned char location,
            unsigned char charmap[] )
```

Creates a character.

**Parameters**

| | |
|---|---|
| *location* | position in the character table |
| *charmap* | character design |

Definition at line 69 of file lcd.c.

References LCDText_WriteByte(), LCDText_WriteCmd(), and WAIT_ChronoUs().

Referenced by gameInit(), and LCDText_Init().

#### 6.22.2.3 LCDText_CursorOff()

```
void LCDText_CursorOff ( )
```

Turns on display cursor off.

Definition at line 148 of file lcd.c.

References LCDText_CMD_DISPLAY_ON, and LCDText_WriteCmd().

Referenced by saveUser(), and Time_changeRoutine().

#### 6.22.2.4 LCDText_CursorOn()

```
void LCDText_CursorOn ( )
```

Turns on display cursor on.

Definition at line 144 of file lcd.c.

References LCDText_CMD_DISPLAY_ON, and LCDText_WriteCmd().

Referenced by saveUser(), and Time_changeRoutine().

#### 6.22.2.5 LCDText_Init()

```
void LCDText_Init (
            void  )
```

Initiates the system to allow access to the LCD peripheral, using 2 lines with 16 columns and 4-bit communication.

Definition at line 45 of file lcd.c.

References DB4, EN, LCDText_Clear(), LCDText_CMD_DISPLAY_OFF, LCDText_CMD_DISPLAY_ON, LCDText↩_CMD_ENTRY_MODE_SET, LCDText_CMD_FUNCTION_SET, LCDText_CreateChar(), LCDText_Locate(), LC↩DText_WriteCmd(), LCDText_WriteNibble(), RS, and WAIT_ChronoUs().

Referenced by main().

#### 6.22.2.6 LCDText_Locate()

```
void LCDText_Locate (
            int row,
            int column )
```

Positions the cursor on the row and column of the display.

Coloca o cursor na posicao (x,y)

**Parameters**

| | |
|---|---|
| *row* | from 1 to 2 |
| *column* | from 0 to 15 |

Definition at line 114 of file lcd.c.

References LCDText_CMD_SET_DDRAM_ADDR, LCDText_COLUMNS, LCDText_LINE_OFFSET, LCDText_LI↩NES, LCDText_WriteCmd(), x, and y.

Referenced by gameRoutine(), LCDText_Clear(), LCDText_Init(), LCDText_WriteString(), saveUser(), Time_↩changeRoutine(), update_DateTimeDisplay(), and updateGameDesign().

#### 6.22.2.7 LCDText_Printf()

```
void LCDText_Printf (
             char * fmt,
              ...  )
```

Writes the string fmt at the current cursor position.

**Parameters**

| | |
|---|---|
| *fmt* | format of the string |

Definition at line 135 of file lcd.c.

References LCDText_WriteString().

Referenced by gameRoutine(), and playerScoresShowDown().

#### 6.22.2.8 LCDText_WriteByte()

```
void LCDText_WriteByte (
             char data,
             unsigned int rs )
```

Definition at line 40 of file lcd.c.

References LCDText_WriteNibble().

Referenced by LCDText_CreateChar(), LCDText_WriteChar(), and LCDText_WriteCmd().

#### 6.22.2.9 LCDText_WriteChar()

```
void LCDText_WriteChar (
             char ch )
```

Writes a character at the current cursor position.

Definition at line 86 of file lcd.c.

References LCDText_DATA, LCDText_WriteByte(), WAIT_ChronoUs(), and x.

Referenced by LCDText_WriteString(), and saveUser().

#### 6.22.2.10 LCDText_WriteCmd()

```
void LCDText_WriteCmd (
            char cmd )
```

Gives an instruction to the LCD controller.

Definition at line 81 of file lcd.c.

References LCDText_CMD, LCDText_WriteByte(), and WAIT_ChronoUs().

Referenced by LCDText_Clear(), LCDText_CreateChar(), LCDText_CursorOff(), LCDText_CursorOn(), LCDText_↩ Init(), and LCDText_Locate().

#### 6.22.2.11 LCDText_WriteNibble()

```
void LCDText_WriteNibble (
            char data,
            unsigned int rs )
```

Definition at line 23 of file lcd.c.

References DB4, EN, RS, and WAIT_ChronoUs().

Referenced by LCDText_Init(), and LCDText_WriteByte().

#### 6.22.2.12 LCDText_WriteString()

```
void LCDText_WriteString (
            char * str )
```

Writes a string at the current cursor position.

Definition at line 92 of file lcd.c.

References LCDText_COLUMNS, LCDText_LINES, LCDText_Locate(), LCDText_WriteChar(), WAIT_ChronoUs(), x, and y.

Referenced by gameRoutine(), LCDText_Printf(), Menu(), playerScoresShowDown(), saveUser(), Time_change↩ Routine(), update_DateTimeDisplay(), and updateGameDesign().

### 6.22.3 Variable Documentation

#### 6.22.3.1 x

```
int x
```

Definition at line 21 of file lcd.c.

Referenced by LCDText_Locate(), LCDText_WriteChar(), and LCDText_WriteString().

#### 6.22.3.2 y

```
int y
```

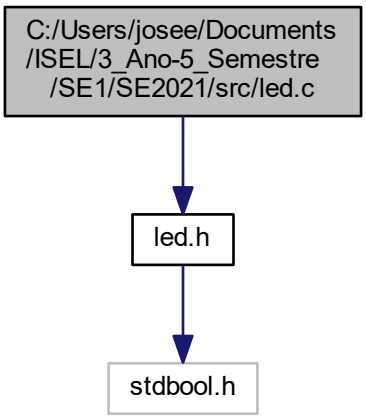
Definition at line 21 of file lcd.c.

Referenced by LCDText_Locate(), and LCDText_WriteString().

## 6.23 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/SE2021/src/led.c File Reference

```
#include "led.h"
```

Include dependency graph for led.c:



## Functions

- void **LED_Init** (bool state)

  *Initiates the system to allow manipulation of the status LED that exists on the LPCXpresso LPC1769 prototyping board.*
- bool **LED_GetState** (void)
- void **LED_On** (void)

  *Turns on the LED.*
- void **LED_Off** (void)

  *Turns off the LED.*

## Variables

- bool **led_state**

## 6.23.1 Function Documentation

### 6.23.1.1 LED_GetState()

```
bool LED_GetState (
             void  )
```

**Returns**

Returns true if the LED is on and false if the LED is off.

Definition at line 26 of file led.c.

References led_state.

### 6.23.1.2 LED_Init()

```
void LED_Init (
             bool state )
```

Initiates the system to allow manipulation of the status LED that exists on the LPCXpresso LPC1769 prototyping board.

**Parameters**

| | |
|---|---|
| *state* | Leave the LED off when the status is false or lit when true. |

Definition at line 16 of file led.c.

References led_state.

#### 6.23.1.3 LED_Off()

```
void LED_Off (
             void  )
```

Turns off the LED.

Definition at line 35 of file led.c.

References led_state.

#### 6.23.1.4 LED_On()

```
void LED_On (
             void  )
```

Turns on the LED.

Definition at line 30 of file led.c.

References led_state.

### 6.23.2 Variable Documentation

#### 6.23.2.1 led_state

```
bool led_state
```

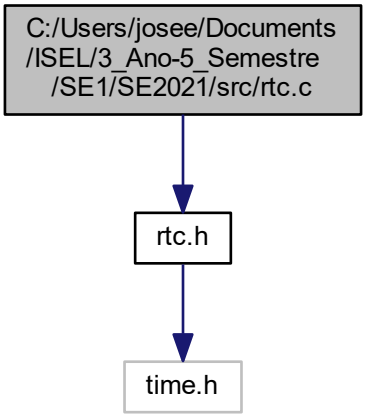
Definition at line 14 of file led.c.

Referenced by LED_GetState(), LED_Init(), LED_Off(), and LED_On().

## 6.24 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/SE2021/src/rtc.c File Reference

```
#include "rtc.h"
```

Include dependency graph for rtc.c:




### Functions

- void **RTC_Init** (time_t seconds)

  *Initiates the system to allow access to the RTC peripheral.*
- void **RTC_GetValue** (struct tm ∗dateTime)
- void **RTC_SetValue** (struct tm ∗dateTime)

  *Updates RTC.*
- void **RTC_SetSeconds** (time_t seconds)

  *Performs the RTC update with the value of the seconds parameter.*
- time_t **RTC_GetSeconds** (void)

### 6.24.1 Function Documentation

#### 6.24.1.1 RTC_GetSeconds()

```
time_t RTC_GetSeconds (
            void  )
```

**Returns**

Returns the current value of the RTC, in seconds since 00:00:00 UTC on January 1 1970

Definition at line 64 of file rtc.c.

References RTC_GetValue().

Referenced by routineChooser().

#### 6.24.1.2 RTC_GetValue()

```
void RTC_GetValue (
            struct tm * dateTime )
```

**Parameters**

| | | |
|---|---|---|
| out | *dateTime* | current RTC value ajusted to 00:00:00 UTC from 1 January 1970. |

Definition at line 25 of file rtc.c.

Referenced by main(), and RTC_GetSeconds().

#### 6.24.1.3 RTC_Init()

```
void RTC_Init (
            time_t seconds )
```

Initiates the system to allow access to the RTC peripheral.

**Parameters**

| | |
|---|---|
| *seconds* | The RTC is started with this value, which represents the seconds since 00:00:00 UTC from 1 January 1970. |

Definition at line 14 of file rtc.c.

References RTC_SetSeconds().

Referenced by main().

#### 6.24.1.4 RTC_SetSeconds()

```
void RTC_SetSeconds (
            time_t seconds )
```

Performs the RTC update with the value of the seconds parameter.

**Parameters**

| | |
|---|---|
| *seconds* | represents the seconds since 00:00:00 UTC of 1 January 1970. |

Definition at line 59 of file rtc.c.

References RTC_SetValue().

Referenced by RTC_Init().

#### 6.24.1.5 RTC_SetValue()

```
void RTC_SetValue (
            struct tm * dateTime )
```

Updates RTC.

**Parameters**

| | |
|---|---|
| *dateTime* | value that is set to update RTC. |

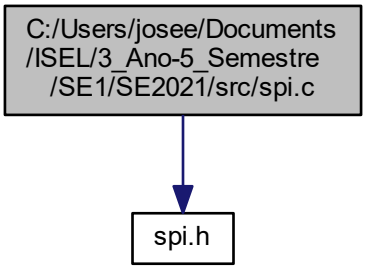
Definition at line 45 of file rtc.c.

Referenced by RTC_SetSeconds(), and Time_changeRoutine().

## 6.25 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/SE2021/src/spi.c File Reference

```
#include "spi.h"
```

Include dependency graph for spi.c:



## Functions

- void **SPI_Init** (void)

  *Faz a iniciação do controlador e configura os respetivos pinos.*
- void **SPI_ConfigTransfer** (int frequency, int bitData, int mode)

  *Configures the transfer.*
- int **SPI_Transfer** (unsigned short ∗txBuffer, unsigned short ∗rxBuffer, int lenght)

  *Makes a transfer. Returns the success or error in the transfer.*

### 6.25.1 Function Documentation

#### 6.25.1.1 SPI_ConfigTransfer()

```
void SPI_ConfigTransfer (
            int frequency,
            int bitData,
            int mode )
```

Configures the transfer.

**Parameters**

| | |
|---|---|
| *frequency* | send/reception frequency |
| *bitData* | number of bits of changed data |
| *mode* | transfer mode is a two bit value (1st bit = CPOL, 2nd bit = CPHA). |

Definition at line 15 of file spi.c.

Referenced by ADXL345_Init().

#### 6.25.1.2 SPI_Init()

```
void SPI_Init (
            void  )
```

Faz a iniciação do controlador e configura os respetivos pinos.

Definition at line 7 of file spi.c.

Referenced by main().

#### 6.25.1.3 SPI_Transfer()

```
int SPI_Transfer (
            unsigned short * txBuffer,
            unsigned short * rxBuffer,
            int lenght )
```

Makes a transfer. Returns the success or error in the transfer.

**Parameters**

| | |
|---|---|
| *txBuffer* | transfer ruffer contains transfer data |
| *rxBuffer* | recieve buffer contains recieved data |
| *length* | how many data is there to be transfered/recieved |

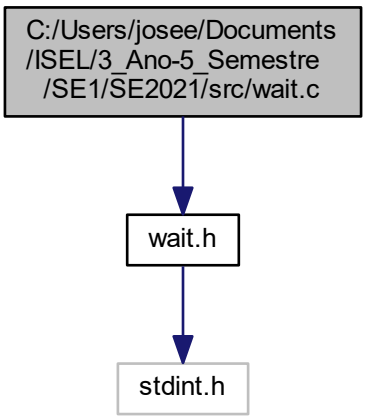
Definition at line 35 of file spi.c.

Referenced by readRegister(), and writeRegister().

## 6.26 C:/Users/josee/Documents/ISEL/3_Ano-5_Semestre/SE1/SE2021/src/wait.c File Reference

```
#include "wait.h"
```

Include dependency graph for wait.c:




### Macros

- #define **SYSTICK_FREQ** (SystemCoreClock / 1000)

### Functions

- void **SysTick_Handler** (void)
- void **WAIT_Milliseconds** (uint32_t millis)

  *Waits a number of milliseconds.*
- int32_t **WAIT_Init** (void)

  *Initialises the wait API for 1 ms resolution.*
- uint32_t **WAIT_GetElapsedMillis** (uint32_t start)

  *Get difference in milliseconds from parameter.*
- void **WAIT_ChronoUs** (uint32_t waitUs)

  *Waits waitUs microseconds.*

## 6.26.1 Macro Definition Documentation

### 6.26.1.1 SYSTICK_FREQ

```
#define SYSTICK_FREQ (SystemCoreClock / 1000)
```

Definition at line 15 of file wait.c.

## 6.26.2 Function Documentation

### 6.26.2.1 SysTick_Handler()

```
void SysTick_Handler (
             void  )
```

Definition at line 19 of file wait.c.

# Index