NPSC

1

Generated by Doxygen 1.8.11

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

File Index

2.1 File List

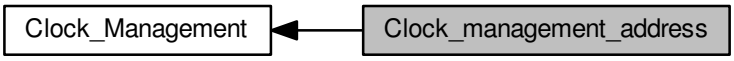
Here is a list of all files with brief descriptions:

Chapter 3

Module Documentation

3.1 Clock_management_address

Collaboration diagram for Clock_management_address:



Macros

- #define TIME_BASE_ADDRESS 0x00
- #define DATE_BASE_ADDRESS 0x04
- #define ALARM_BASE_ADDRESS 0x08

3.1.1 Detailed Description

3.1.2 Macro Definition Documentation

3.1.2.1 #define ALARM_BASE_ADDRESS 0x08

Definition at line 43 of file clock_management.h.

3.1.2.2 #define DATE_BASE_ADDRESS 0x04

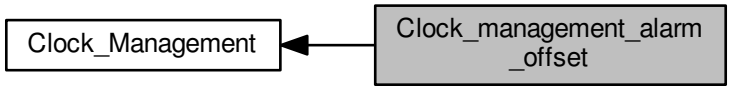
Definition at line 42 of file clock_management.h.

3.1.2.3 #define TIME_BASE_ADDRESS 0x00

Definition at line 41 of file clock_management.h.

3.2 Clock_management_alarm_offset

Collaboration diagram for Clock_management_alarm_offset:



Macros

- #define OFFSET_NAME 0x00
- #define OFFSET_DATEWEEKDAY NAME_SIZE
- #define OFFSET_DATEWEEKDAY_SEL OFFSET_DATEWEEKDAY + 1
- #define OFFSET_MASK OFFSET_DATEWEEKDAY_SEL + 4
- #define OFFSET_H12 OFFSET_MASK + 4
- #define OFFSET_HOURS OFFSET_H12 + 1
- #define OFFSET_MINUTES OFFSET_HOURS + 1
- #define OFFSET_SECONDS OFFSET_MINUTES + 1

3.2.1 Detailed Description

3.2.2 Macro Definition Documentation

3.2.2.1 #define OFFSET_DATEWEEKDAY NAME_SIZE

Definition at line 52 of file clock_management.h.

3.2.2.2 #define OFFSET_DATEWEEKDAY_SEL OFFSET_DATEWEEKDAY + 1

Definition at line 53 of file clock_management.h.

3.2.2.3 #define OFFSET_H12 OFFSET_MASK + 4

Definition at line 55 of file clock_management.h.

3.2.2.4 #define OFFSET_HOURS OFFSET_H12 + 1

Definition at line 56 of file clock_management.h.

3.2.2.5 #define OFFSET_MASK OFFSET_DATEWEEKDAY_SEL + 4

Definition at line 54 of file clock_management.h.

3.2.2.6 #define OFFSET_MINUTES OFFSET_HOURS + 1

Definition at line 57 of file clock_management.h.

3.2.2.7 #define OFFSET_NAME 0x00

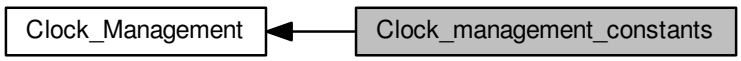
Definition at line 51 of file clock_management.h.

3.2.2.8 #define OFFSET_SECONDS OFFSET_MINUTES + 1

Definition at line 58 of file clock_management.h.

3.3 Clock_management_constants

Collaboration diagram for Clock_management_constants:



Macros

- #define NAME_SIZE 31

3.3.1 Detailed Description

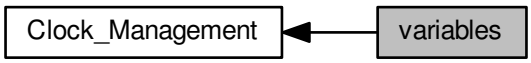
3.3.2 Macro Definition Documentation

3.3.2.1 #define NAME_SIZE 31

Definition at line 66 of file clock_management.h.

3.4 variables

Collaboration diagram for variables:



Variables

- uint16_t eeprom_index
- uint16_t next_alarm

3.4.1 Detailed Description

3.4.2 Variable Documentation

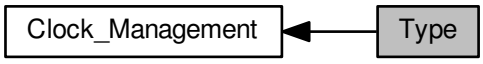
3.4.2.1 uint16_t eeprom_index

3.4.2.2 uint16_t next_alarm

3.5 Type

Alarm type.

Collaboration diagram for Type:



Data Structures

- struct Alarm_Definition

3.5.1 Detailed Description

Alarm type.

3.5.2 Data Structure Documentation

3.5.2.1 struct Alarm_Definition

Definition at line 86 of file clock_management.h.

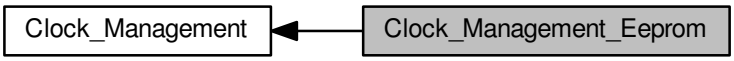
Data Fields

char	alarmName[NAME_SIZE]	
RTC_AlarmTypeDef	alarmParameters	

3.6 Clock_Management_Eeprom

Saving and Loading of clock information.

Collaboration diagram for Clock_Management_Eeprom:



Functions

- ErrorStatus ClockManagement_saveAlarm (Alarm_Definition *Alarm_Def, uint16_t address)
  Save an alarm settings to eeprom.
- ErrorStatus ClockManagement_saveTime (RTC_TimeTypeDef *Time_Def)
  Save the time settings to eeprom.
- ErrorStatus ClockManagement_saveDate (RTC_DateTypeDef *Date_Def)
  Save the date settings to eeprom.
- Alarm_Definition ClockManagement_loadAlarm (uint16_t index)
  load an alarm settings from eeprom
- RTC_TimeTypeDef ClockManagement_loadTime (void)
  Load the time settings from eeprom.
- RTC_DateTypeDef ClockManagement_loadDate (void)
  Load the date settings from eeprom.

3.6.1 Detailed Description

Saving and Loading of clock information.

```
================================================================================
       ##### Clock Management: Eeprom date, time, and alarm access #####
================================================================================
```

3.6.2 Function Documentation

3.6.2.1 Alarm_Definition ClockManagement_loadAlarm (uint16_t *index*)

load an alarm settings from eeprom

Note

Load in this order: Name, DateWeekDay, DateWeekDaySel, Mask H12, Hours, Minutes, Seconds

Parameters

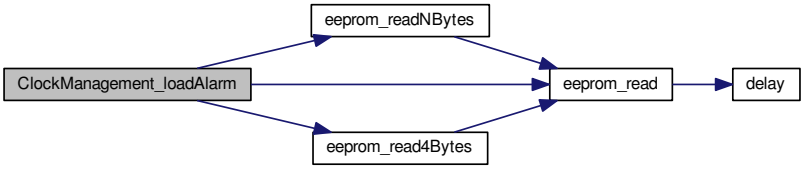
index	the index in memory of the alarm

Return values

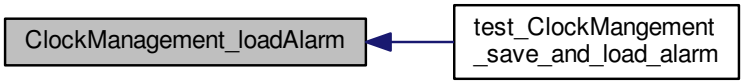
Alarm_Definition	

Definition at line 103 of file clock_management.c.

Here is the call graph for this function:



Here is the caller graph for this function:



3.6.2.2 RTC_DateTypeDef ClockManagement_loadDate (void)

Load the date settings from eeprom.

Note

Load Date, Month, WeekDay, and Year in that order

Parameters

None	

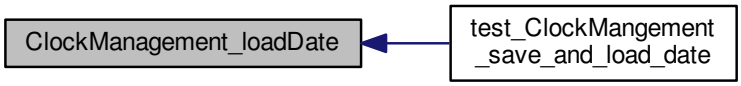
Return values

RTC_DateTypeDef	

Definition at line 139 of file clock_management.c.

Here is the call graph for this function:

Here is the caller graph for this function:



3.6.2.3 RTC_TimeTypeDef ClockManagement_loadTime (void)

Load the time settings from eeprom.

Note

Load H12, Hours, Minutes, and Seconds in that order

Parameters

None	

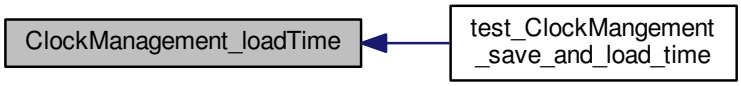
Return values

RTC_TimeTypeDef	

Definition at line 123 of file clock_management.c.

Here is the call graph for this function:

Here is the caller graph for this function:



3.6.2.4 ErrorStatus ClockManagement_saveAlarm (Alarm_Definition * *Alarm_Def,* uint16_t *address*)

Save an alarm settings to eeprom.

Note

Save in this order: Name, DateWeekDat, DateWeekDaySel, Mask H12, Hours, Minutes, Seconds

Parameters

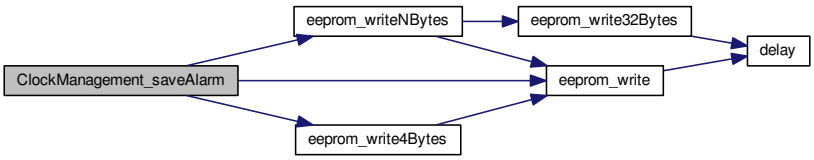
Alarm_Def	the alarm setitngs

Return values

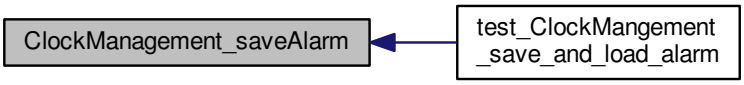
ErrorStatus	

Definition at line 57 of file clock_management.c.

Here is the call graph for this function:



Here is the caller graph for this function:



3.6.2.5 ErrorStatus ClockManagement_saveDate (RTC_DateTypeDef * *Date_Def*)

Save the date settings to eeprom.

Note

Save Date, Month, WeekDay, and Year in that order

Parameters

Date_Def	the date setitngs

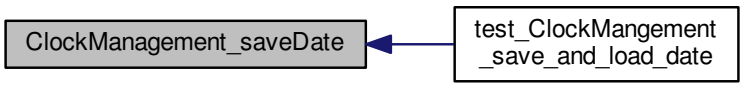
Return values

ErrorStatus	

Definition at line 89 of file clock_management.c.

Here is the call graph for this function:

Here is the caller graph for this function:



3.6.2.6 ErrorStatus ClockManagement_saveTime (RTC_TimeTypeDef * *Time_Def*)

Save the time settings to eeprom.

Note

Save H12, Hours, Minutes, and Seconds in that order

Parameters

Time_Def	the time setitngs

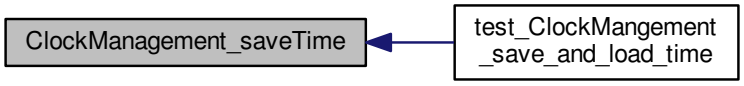
Return values

ErrorStatus	

Definition at line 75 of file clock_management.c.

Here is the call graph for this function:

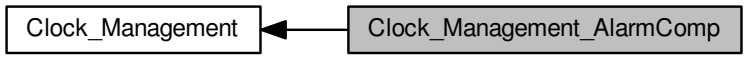
Here is the caller graph for this function:



3.7 Clock_Management_AlarmComp

Comparison between dates, times and alarms.

Collaboration diagram for Clock_Management_AlarmComp:



Functions

- bool ClockManagement_isTimeBefore (RTC_TimeTypeDef ∗time1, RTC_TimeTypeDef ∗time2)
  Compare two time.
- bool ClockManagement_isDateBefore (RTC_DateTypeDef ∗date1, RTC_DateTypeDef ∗date2)
  Convert an date to an integer.
- bool ClockManagement_isAlarmBefore (Alarm_Definition ∗alarm1, Alarm_Definition ∗alarm2)
  Compare two alarm.

3.7.1 Detailed Description

Comparison between dates, times and alarms.

```
================================================================================
       ##### Clock Management: Time, Date, and Alarm comparison #####
================================================================================
```

3.7.2 Function Documentation

3.7.2.1 bool ClockManagement_isAlarmBefore (Alarm_Definition ∗ *alarm1,* Alarm_Definition ∗ *alarm2*)

Compare two alarm.

Note

Only support same dateWeekDaySel comparison TODO include mask comparison

Parameters

is	alarm1 before alarm2?

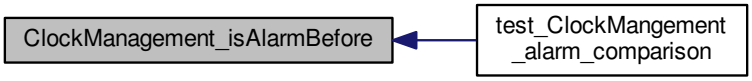
Return values

uint32_t	representing the time

Definition at line 212 of file clock_management.c.

Here is the call graph for this function:

Here is the caller graph for this function:



3.7.2.2 bool ClockManagement_isDateBefore (RTC_DateTypeDef * *date1*, RTC_DateTypeDef * *date2*)

Convert an date to an integer.

Parameters

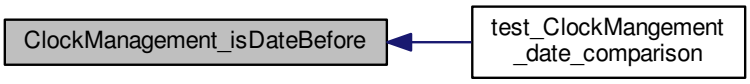
is	date1 before date2?

Return values

bool	

Definition at line 192 of file clock_management.c.

Here is the caller graph for this function:



3.7.2.3 bool ClockManagement_isTimeBefore (RTC_TimeTypeDef * *time1*, RTC_TimeTypeDef * *time2*)

Compare two time.

Parameters

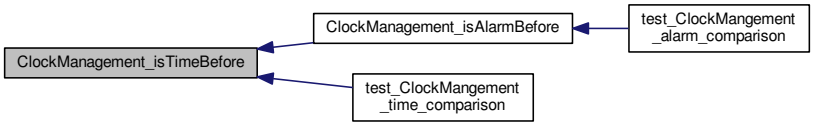
is	time1 before time2?

Return values

bool	

Definition at line 170 of file clock_management.c.

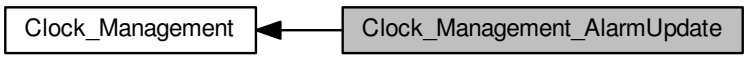
Here is the caller graph for this function:



3.8 Clock_Management_AlarmUpdate

Manages updates of alarms and alarm parameters.

Collaboration diagram for Clock_Management_AlarmUpdate:



Functions

- void CLockManagement_updateAlarm (void)

  Update the alarm with the closest alarm.

3.8.1 Detailed Description

Manages updates of alarms and alarm parameters.

```
================================================================================
            ##### Clock Management: Updates of alarm  #####
================================================================================
```

3.8.2 Function Documentation

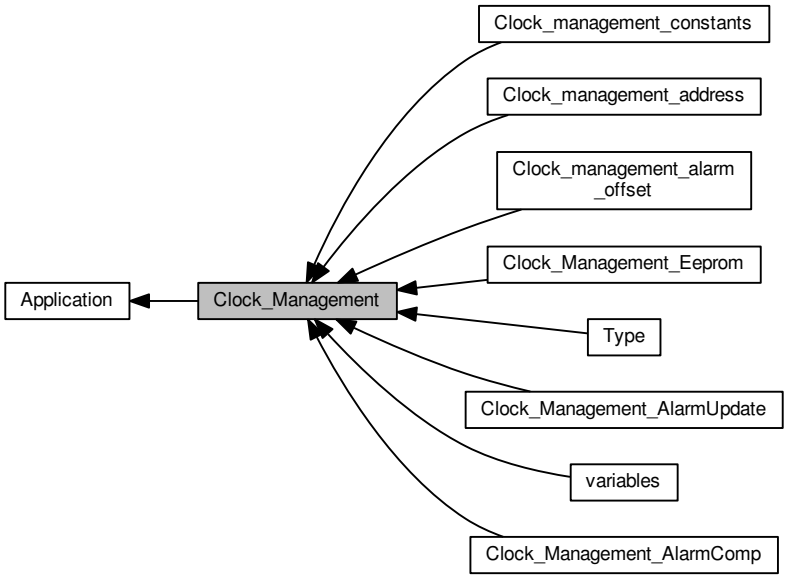
3.8.2.1 void CLockManagement_updateAlarm (void)

Update the alarm with the closest alarm.

Definition at line 242 of file clock_management.c.

3.9 Clock_Management

Collaboration diagram for Clock_Management:



Modules

- Clock_management_address
- Clock_management_alarm_offset
- Clock_management_constants
- variables
- Type

  Alarm type.
- Clock_Management_Eeprom

  Saving and Loading of clock information.
- Clock_Management_AlarmComp

  Comparison between dates, times and alarms.
- Clock_Management_AlarmUpdate

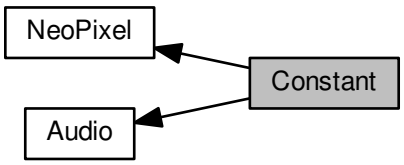
  Manages updates of alarms and alarm parameters.

3.9.1 Detailed Description

3.10 Constant

Define audio frequency and DMA frequency.

Collaboration diagram for Constant:



Macros

- #define AUDIO_FREQUENCY 11000
- #define DMA_FREQUENCY (86000000/(2∗AUDIO_FREQUENCY))
- #define WS2812_FREQ (8E5)
- #define TIMER_CLOCK_FREQ (84E6)
- #define TIMER_PERIOD (TIMER_CLOCK_FREQ / WS2812_FREQ)
- #define LED_NUMBER (4)
- #define LED_DATA_SIZE (LED_NUMBER ∗ 24)
- #define RESET_SLOTS_BEGIN (50)
- #define RESET_SLOTS_END (50)
- #define WS2812_LAST_SLOT (1)
- #define LED_BUFFER_SIZE (RESET_SLOTS_BEGIN + LED_DATA_SIZE + WS2812_LAST_SLOT + R↩ ESET_SLOTS_END)
- #define WS2812_0 (TIMER_PERIOD / 3)
- #define WS2812_1 (TIMER_PERIOD ∗ 2 / 3)
- #define WS2812_RESET (0)
- #define MAX_8BIT (255)

3.10.1 Detailed Description

Define audio frequency and DMA frequency.

Defines constants.

3.10.2 Macro Definition Documentation

3.10.2.1 #define AUDIO_FREQUENCY 11000

Definition at line 43 of file NPC_audio.h.

3.10.2.2 #define DMA_FREQUENCY (86000000/(2∗AUDIO_FREQUENCY))

Definition at line 44 of file NPC_audio.h.

3.10.2.3 #define LED_BUFFER_SIZE (RESET_SLOTS_BEGIN + LED_DATA_SIZE + WS2812_LAST_SLOT + RESET_SLOTS_END)

Definition at line 50 of file NPC_neopixel.h.

3.10.2.4 #define LED_DATA_SIZE (LED_NUMBER ∗ 24)

Definition at line 46 of file NPC_neopixel.h.

3.10.2.5 #define LED_NUMBER (4)

Definition at line 45 of file NPC_neopixel.h.

3.10.2.6 #define MAX_8BIT (255)

Definition at line 54 of file NPC_neopixel.h.

3.10.2.7 #define RESET_SLOTS_BEGIN (50)

Definition at line 47 of file NPC_neopixel.h.

3.10.2.8 #define RESET_SLOTS_END (50)

Definition at line 48 of file NPC_neopixel.h.

3.10.2.9 #define TIMER_CLOCK_FREQ (84E6)

Definition at line 43 of file NPC_neopixel.h.

3.10.2.10 #define TIMER_PERIOD (TIMER_CLOCK_FREQ / WS2812_FREQ)

Definition at line 44 of file NPC_neopixel.h.

3.10.2.11 #define WS2812_0 (TIMER_PERIOD / 3)

Definition at line 51 of file NPC_neopixel.h.

3.10.2.12 #define WS2812_1 (TIMER_PERIOD * 2 / 3)

Definition at line 52 of file NPC_neopixel.h.

3.10.2.13 #define WS2812_FREQ (8E5)

Definition at line 42 of file NPC_neopixel.h.

3.10.2.14 #define WS2812_LAST_SLOT (1)

Definition at line 49 of file NPC_neopixel.h.

3.10.2.15 #define WS2812_RESET (0)

Definition at line 53 of file NPC_neopixel.h.

3.11 Configuration functions

Audio configuration functions.

Collaboration diagram for Configuration functions:

Functions

- void audio_disable (void)
  Disable the DMA.
- void audio_init (uint16_t ∗DACBuffer, uint16_t Size)
  Perform audio initialization.

3.11.1 Detailed Description

Audio configuration functions.

3.11.2 Function Documentation

3.11.2.1 void audio_disable (void)

Disable the DMA.

Parameters

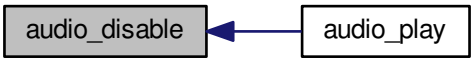
None	

Return values

None	

Definition at line 48 of file NPC_audio.c.

Here is the caller graph for this function:



3.11.2.2 void audio_init (uint16_t * *DACBuffer*, uint16_t *Size*)

Perform audio initialization.

Parameters

DACBuffer	Array to be pushed to the DMA
Mode	DMA Mode (default:DMA_Mode_Normal)
Size	sample size (default:SAMPLE_SIZE)

Return values

None	

Definition at line 61 of file NPC_audio.c.

Here is the caller graph for this function:

3.12 Play audio functions

Audio functions.

Collaboration diagram for Play audio functions:

Functions

- void audio_play (uint16_t *DACBuffer, uint16_t Size)
  Play a sample.

3.12.1 Detailed Description

Audio functions.

3.12.2 Function Documentation

3.12.2.1 void audio_play (uint16_t * *DACBuffer,* uint16_t *Size*)

Play a sample.

Parameters

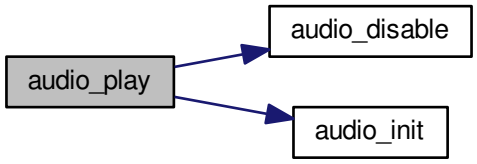
DACBuffer	Array to be pushed to the DMA
Size	sample size (default:SAMPLE_SIZE)

Returns

None

Definition at line 136 of file NPC_audio.c.

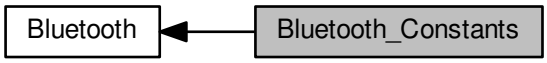
Here is the call graph for this function:



3.13 Bluetooth_Constants

define bluetooth constant

Collaboration diagram for Bluetooth_Constants:



Macros

- #define BLUETOOTH_PERIPH_USARTX RCC_APB2Periph_USART1
- #define BLUETOOTH_PERIPH_GPIOX RCC_AHB1Periph_GPIOB
- #define BLUETOOTH_GPIOX GPIOB
- #define BLUETOOTH_TX_PIN GPIO_Pin_6
- #define BLUETOOTH_RX_PIN GPIO_Pin_7
- #define BLUETOOTH_TX_PINSOURCE GPIO_PinSource6
- #define BLUETOOTH_RX_PINSOURCE GPIO_PinSource7
- #define BLUETOOTH_AF_USART GPIO_AF_USART1
- #define BLUETOOTH_USARTX USART1
- #define BLUETOOTH_USARTX_IRQ USART1_IRQn
- #define BLUETOOTH_BAUDRATE 9600

3.13.1 Detailed Description

define bluetooth constant

3.13.2 Macro Definition Documentation

3.13.2.1 #define BLUETOOTH_AF_USART GPIO_AF_USART1

Definition at line 48 of file NPC_bluetooth.h.

3.13.2.2 #define BLUETOOTH_BAUDRATE 9600

Definition at line 51 of file NPC_bluetooth.h.

3.13.2.3 #define BLUETOOTH_GPIOX GPIOB

Definition at line 43 of file NPC_bluetooth.h.

3.13.2.4 #define BLUETOOTH_PERIPH_GPIOX RCC_AHB1Periph_GPIOB

Definition at line 42 of file NPC_bluetooth.h.

3.13.2.5 #define BLUETOOTH_PERIPH_USARTX RCC_APB2Periph_USART1

Definition at line 41 of file NPC_bluetooth.h.

3.13.2.6 #define BLUETOOTH_RX_PIN GPIO_Pin_7

Definition at line 45 of file NPC_bluetooth.h.

3.13.2.7 #define BLUETOOTH_RX_PINSOURCE GPIO_PinSource7

Definition at line 47 of file NPC_bluetooth.h.

3.13.2.8 #define BLUETOOTH_TX_PIN GPIO_Pin_6

Definition at line 44 of file NPC_bluetooth.h.

3.13.2.9 #define BLUETOOTH_TX_PINSOURCE GPIO_PinSource6

Definition at line 46 of file NPC_bluetooth.h.

3.13.2.10 #define BLUETOOTH_USARTX USART1

Definition at line 49 of file NPC_bluetooth.h.

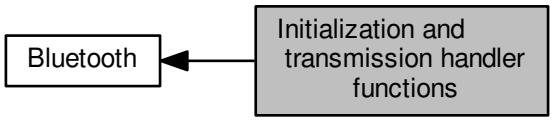
3.13.2.11 #define BLUETOOTH_USARTX_IRQ USART1_IRQn

Definition at line 50 of file NPC_bluetooth.h.

3.14 Initialization and transmission handler functions

bluetooth initialization functions

Collaboration diagram for Initialization and transmission handler functions:



Functions

- void bluetooth_init (void)

  Initialize the bluetooth and set baudrate to 9600.
- void USART1_IRQHandler (void)

  Global interrupt handler for USART1.
- void DMA2_Stream5_IRQHandler (void)

  Global interrupt handler for DMA2 stream5.
- void bluetooth_buffer_update (void)

3.14.1 Detailed Description

bluetooth initialization functions

3.14.2 Function Documentation

3.14.2.1 void bluetooth_buffer_update (void)

MUST BE A TASK Loop data back to UART data register

Definition at line 120 of file NPC_bluetooth.c.

3.14.2.2 void bluetooth_init (void)

Initialize the bluetooth and set baudrate to 9600.

Parameters

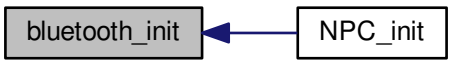
None	

Return values

None	

Definition at line 47 of file NPC_bluetooth.c.

Here is the caller graph for this function:



3.14.2.3 void DMA2_Stream5_IRQHandler (void)

Global interrupt handler for DMA2 stream5.

Note

Except memcpy, there is no functions used to

Transfer could be completed by 2 events:

- All data actually transfered (NDTR = 0)
- Stream disabled inside USART IDLE line detected interrupt (NDTR != 0)

Definition at line 156 of file NPC_bluetooth.c.

3.14.2.4 void USART1_IRQHandler (void)

Global interrupt handler for USART1.

Definition at line 138 of file NPC_bluetooth.c.

3.15 Transmission functions

bluetooth transmission functions

Collaboration diagram for Transmission functions:

Functions

- void bluetooth_send (uint8_t *data)
  send string to the hc-06

3.15.1 Detailed Description

bluetooth transmission functions

3.15.2 Function Documentation

3.15.2.1 void bluetooth_send (uint8_t * *data*)

send string to the hc-06

Parameters

data	string to be sent

Return values

None	

Definition at line 217 of file NPC_bluetooth.c.

3.16 RTC_PREDIV_Definitions

definition of prescaler for Asynchronous and Synchronous

Collaboration diagram for RTC_PREDIV_Definitions:

Macros

- #define RTC_PREDIV_A 0x7C
- #define RTC_PREDIV_S 0X1F3F

3.16.1 Detailed Description

definition of prescaler for Asynchronous and Synchronous

3.16.2 Macro Definition Documentation

3.16.2.1 #define RTC_PREDIV_A 0x7C

Definition at line 43 of file NPC_clock.h.

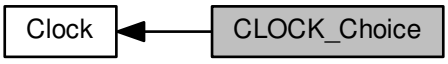
3.16.2.2 #define RTC_PREDIV_S 0X1F3F

Definition at line 44 of file NPC_clock.h.

3.17 CLOCK_Choice

Clock A or B.

Collaboration diagram for CLOCK_Choice:



Macros

- #define CLOCK_A RTC_Alarm_A
- #define CLOCK_B RTC_Alarm_B

3.17.1 Detailed Description

Clock A or B.

3.17.2 Macro Definition Documentation

3.17.2.1 #define CLOCK_A RTC_Alarm_A

Definition at line 54 of file NPC_clock.h.

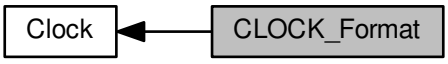
3.17.2.2 #define CLOCK_B RTC_Alarm_B

Definition at line 55 of file NPC_clock.h.

3.18 CLOCK_Format

AM or PM.

Collaboration diagram for CLOCK_Format:



Macros

- #define AM RTC_H12_AM
- #define PM RTC_H12_PM

3.18.1 Detailed Description

AM or PM.

3.18.2 Macro Definition Documentation

3.18.2.1 #define AM RTC_H12_AM

Definition at line 65 of file NPC_clock.h.

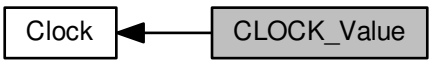
3.18.2.2 #define PM RTC_H12_PM

Definition at line 66 of file NPC_clock.h.

3.19 CLOCK_Value

Access time or date parameters.

Collaboration diagram for CLOCK_Value:



Macros

- #define CLOCK_WeekDay (uint8_t) (clock_getDate() >> 24)
- #define CLOCK_Date (uint8_t) (clock_getDate() >> 16)
- #define CLOCK_Month (uint8_t) (clock_getDate() >> 8)
- #define CLOCK_Year (uint8_t) clock_getDate()
- #define CLOCK_Hours (uint8_t) (clock_getTime() >> 24)
- #define CLOCK_Minutes (uint8_t) (clock_getTime() >> 16)
- #define CLOCK_Seconds (uint8_t) (clock_getTime() >> 8)
- #define CLOCK_Format (uint8_t) clock_getTime()

3.19.1 Detailed Description

Access time or date parameters.

3.19.2 Macro Definition Documentation

3.19.2.1 #define CLOCK_Date (uint8_t) (clock_getDate() >> 16)

Definition at line 76 of file NPC_clock.h.

3.19.2.2 #define CLOCK_Format (uint8_t) clock_getTime()

Definition at line 82 of file NPC_clock.h.

3.19.2.3 #define CLOCK_Hours (uint8_t) (clock_getTime() >> 24)

Definition at line 79 of file NPC_clock.h.

3.19.2.4 #define CLOCK_Minutes (uint8_t) (clock_getTime() >> 16)

Definition at line 80 of file NPC_clock.h.

3.19.2.5 #define CLOCK_Month (uint8_t) (clock_getDate() >> 8)

Definition at line 77 of file NPC_clock.h.

3.19.2.6 #define CLOCK_Seconds (uint8_t) (clock_getTime() >> 8)

Definition at line 81 of file NPC_clock.h.

3.19.2.7 #define CLOCK_WeekDay (uint8_t) (clock_getDate() >> 24)

Definition at line 75 of file NPC_clock.h.

3.19.2.8 #define CLOCK_Year (uint8_t) clock_getDate()

Definition at line 78 of file NPC_clock.h.

3.20 REPEAT_Definitions

Alarm repeat options.

Collaboration diagram for REPEAT_Definitions:

Macros

- #define REPEAT_DateWeekDay (uint32_t) RTC_AlarmMask_DateWeekDay
- #define REPEAT_Hours (uint32_t) RTC_AlarmMask_Hours
- #define REPEAT_Minutes (uint32_t) RTC_AlarmMask_Minutes
- #define REPEAT_Seconds (uint32_t) RTC_AlarmMask_Seconds
- #define REPEAT_None (uint32_t) RTC_AlarmMask_None

3.20.1 Detailed Description

Alarm repeat options.

3.20.2 Macro Definition Documentation

3.20.2.1 #define REPEAT_DateWeekDay (uint32_t) RTC_AlarmMask_DateWeekDay

Definition at line 91 of file NPC_clock.h.

3.20.2.2 #define REPEAT_Hours (uint32_t) RTC_AlarmMask_Hours

Definition at line 92 of file NPC_clock.h.

3.20.2.3 #define REPEAT_Minutes (uint32_t) RTC_AlarmMask_Minutes

Definition at line 93 of file NPC_clock.h.

3.20.2.4 #define REPEAT_None (uint32_t) RTC_AlarmMask_None

Definition at line 95 of file NPC_clock.h.

3.20.2.5 #define REPEAT_Seconds (uint32_t) RTC_AlarmMask_Seconds

Definition at line 94 of file NPC_clock.h.

3.21 Initialisation functions

Clock initialisation functions.

Collaboration diagram for Initialisation functions:

Functions

- void clock_init (void)

  Initialise the clock to 1Hz and setup peripherals for Alarm.

3.21.1 Detailed Description

Clock initialisation functions.

3.21.2 Function Documentation

3.21.2.1 void clock_init (void)

Initialise the clock to 1Hz and setup peripherals for Alarm.

Parameters

None	

Return values

None	

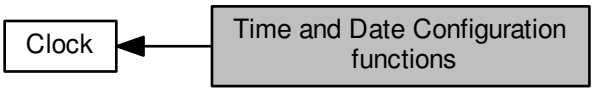
Definition at line 74 of file NPC_clock.c.

Here is the caller graph for this function:

3.22 Time and Date Configuration functions

Clock time and date configuration functions.

Collaboration diagram for Time and Date Configuration functions:



Functions

- ErrorStatus clock_setDate (uint8_t weekDay, uint8_t month, uint8_t date, uint8_t year)
  Set the clock's date.
- ErrorStatus clock_setTime (uint8_t am_pm, uint8_t hours, uint8_t minutes, uint8_t second)
  Set the clock's time.
- uint32_t clock_getDate (void)
  Get the date encoded in a 32b format.
- uint32_t clock_getTime (void)
  Get the time encoded in a 32b format.

3.22.1 Detailed Description

Clock time and date configuration functions.

3.22.2 Function Documentation

3.22.2.1 uint32_t clock_getDate (void)

Get the date encoded in a 32b format.

Parameters

None	

Return values

An	uint32_t containing the weekDay as its MB3, date : MB2, month : MB1, year : MB0

Definition at line 160 of file NPC_clock.c.

3.22.2.2 uint32_t clock_getTime (void)

Get the time encoded in a 32b format.

Parameters

None	

Return values

An	uint32_t containing the hour as its MB3, minutes : MB2, Seconds : MB1, format : MB0

Definition at line 175 of file NPC_clock.c.

3.22.2.3 ErrorStatus clock_setDate (uint8_t *weekDay,* uint8_t *month,* uint8_t *date,* uint8_t *year*)

Set the clock's date.

Parameters

None	

Return values

ErrorStatus	representing the outcome of the operation • SUCCESS: RTC Shift registers are configured • ERROR: RTC Shift registers are not configured

Definition at line 128 of file NPC_clock.c.

Here is the caller graph for this function:

3.22.2.4 ErrorStatus clock_setTime (uint8_t *am_pm,* uint8_t *hours,* uint8_t *minutes,* uint8_t *second*)

Set the clock's time.

Parameters

None	

Return values

ErrorStatus	representing the outcome of the operation • SUCCESS: RTC Shift registers are configured • ERROR: RTC Shift registers are not configured

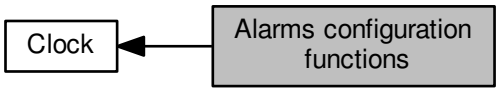
Definition at line 145 of file NPC_clock.c.

Here is the caller graph for this function:

3.23 Alarms configuration functions

Clock alarm configuration functions.

Collaboration diagram for Alarms configuration functions:



Functions

- RTC_AlarmTypeDef clock_createAlarm (uint8_t am_pm, uint8_t hours, uint8_t minutes, uint8_t seconds, uint32_t dateWeekDaySel, uint8_t dateWeekDay, uint32_t repeat)

  Create an Alarm Structure given all the parameters.
- void clock_setA (RTC_AlarmTypeDef ∗Alarm)

  Set an alarm to RTC_Alarm_A, given a Alarm structure RTC_AlarmTypeDef.
- void clock_setAlarm (uint8_t am_pm, uint8_t hours, uint8_t minutes, uint8_t seconds, uint32_t dateWeek↩ DaySel, uint8_t dateWeekDay, uint32_t repeat)

  Set an alarm to RTC_Alarm_A, given all the alarm parameters.
- void RTC_Alarm_IRQHandler (void)

  Alarm Handler.

3.23.1 Detailed Description

Clock alarm configuration functions.

3.23.2 Function Documentation

3.23.2.1 RTC_AlarmTypeDef clock_createAlarm (uint8_t *am_pm,* uint8_t *hours,* uint8_t *minutes,* uint8_t *seconds,* uint32_t *dateWeekDaySel,* uint8_t *dateWeekDay,* uint32_t *repeat*)

Create an Alarm Structure given all the parameters.

Parameters

am_pm	AM PM format (CLOCK_AM)
hours	Alarm hours
minutes	Alarm minutes
seconds	Alarm seconds
dateWeekDaySel	Date of WeekDay selection RTC_AlarmDateWeekDay_Definitions
dateWeekDay	Specify Alarm Date/Weekday if Date then value range from 1-31, else RTC_WeekDay_Definitions
repeat	Specify the repetition of the Alarm

Return values

An	RTC_AlarmTypeDef containing all the parameters above

Definition at line 204 of file NPC_clock.c.

3.23.2.2 void clock_setA (RTC_AlarmTypeDef * *Alarm*)

Set an alarm to RTC_Alarm_A, given a Alarm structure RTC_AlarmTypeDef.

Parameters

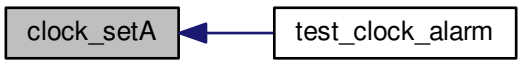
Alarm	A pointer to the RTC_AlarmTypeDef

Return values

None	

Definition at line 225 of file NPC_clock.c.

Here is the caller graph for this function:



3.23.2.3 void clock_setAlarm (uint8_t *am_pm,* uint8_t *hours,* uint8_t *minutes,* uint8_t *seconds,* uint32_t *dateWeekDaySel,* uint8_t *dateWeekDay,* uint32_t *repeat*)

Set an alarm to RTC_Alarm_A, given all the alarm parameters.

Parameters

am_pm	AM PM format (CLOCK_AM)
hours	Alarm hours
minutes	Alarm minutes
seconds	Alarm seconds
dateWeekDaySel	Date of WeekDay selection RTC_AlarmDateWeekDay_Definitions
dateWeekDay	Specify Alarm Date/Weekday if Date then value range from 1-31, else RTC_WeekDay_Definitions
repeat	Specify the repetition of the Alarm

Return values

None	

Definition at line 244 of file NPC_clock.c.

3.23.2.4 void RTC_Alarm_IRQHandler (void)

Alarm Handler.

Parameters

None	

Return values

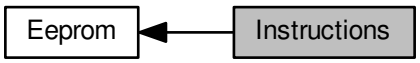
None	

Definition at line 271 of file NPC_clock.c.

3.24 Instructions

25LC640A instruction set

Collaboration diagram for Instructions:



Macros

- #define WREN 0b00000110
- #define WRDI 0b00000100
- #define RDSR 0b00000101
- #define WRSR 0b00000001
- #define READ 0b00000011
- #define WRITE 0b00000010

3.24.1 Detailed Description

25LC640A instruction set

3.24.2 Macro Definition Documentation

3.24.2.1 #define RDSR 0b00000101

Definition at line 44 of file NPC_eeprom.h.

3.24.2.2 #define READ 0b00000011

Definition at line 46 of file NPC_eeprom.h.

3.24.2.3 #define WRDI 0b00000100

Definition at line 43 of file NPC_eeprom.h.

3.24.2.4 #define WREN 0b00000110

Definition at line 42 of file NPC_eeprom.h.

3.24.2.5 #define WRITE 0b00000010

Definition at line 47 of file NPC_eeprom.h.

3.24.2.6 #define WRSR 0b00000001

Definition at line 45 of file NPC_eeprom.h.

3.25 Utilities

Collaboration diagram for Utilities:

Macros

- #define PAGE_LENGTH 32
- #define EEPROM_SIZE 0xFA00

3.25.1 Detailed Description

3.25.2 Macro Definition Documentation

3.25.2.1 #define EEPROM_SIZE 0xFA00

Definition at line 57 of file NPC_eeprom.h.

3.25.2.2 #define PAGE_LENGTH 32

Definition at line 56 of file NPC_eeprom.h.

3.26 Initialisation functions

Eeprom initialisation functions.

Collaboration diagram for Initialisation functions:

Functions

- void eeprom_init (void)

  Initialise communication to the eeprom.

3.26.1 Detailed Description

Eeprom initialisation functions.

3.26.2 Function Documentation

3.26.2.1 void eeprom_init (void)

Initialise communication to the eeprom.

Parameters

None	

Return values

None	

Definition at line 48 of file NPC_eeprom.c.

Here is the caller graph for this function:

3.27 Transmission functions

Eeprom data transmission functions.

Collaboration diagram for Transmission functions:

Functions

- ErrorStatus eeprom_write (uint16_t address, uint8_t data)
  Write a byte to the eeprom.
- uint8_t eeprom_read (uint16_t address)
  Read a byte from the eeprom.
- ErrorStatus eeprom_write32Bytes (uint16_t baseAddress, uint8_t ∗data)
  Write a page to the eeprom.
- uint32_t eeprom_read4Bytes (uint16_t baseAddress)
  Read a 32byte from eeprom.
- ErrorStatus eeprom_writeNBytes (uint16_t baseAddress, uint8_t ∗data, uint16_t N)
  Write N bytes to the eeprom.
- ErrorStatus eeprom_write4Bytes (uint16_t baseAddress, uint8_t ∗data)
  Write 4 bytes to the eeprom.
- void eeprom_readNBytes (uint16_t baseAddress, uint8_t ∗data, uint16_t N)
- void eeprom_clear (void)
  Clear eeprom data.

3.27.1 Detailed Description

Eeprom data transmission functions.

3.27.2 Function Documentation

3.27.2.1 void eeprom_clear (void)

Clear eeprom data.

Parameters

None	

Return values

None	

Definition at line 242 of file NPC_eeprom.c.

Here is the call graph for this function:

3.27.2.2 uint8_t eeprom_read (uint16_t *address*)

Read a byte from the eeprom.

Parameters

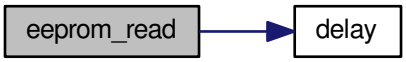
address	The address of the memory

Return values

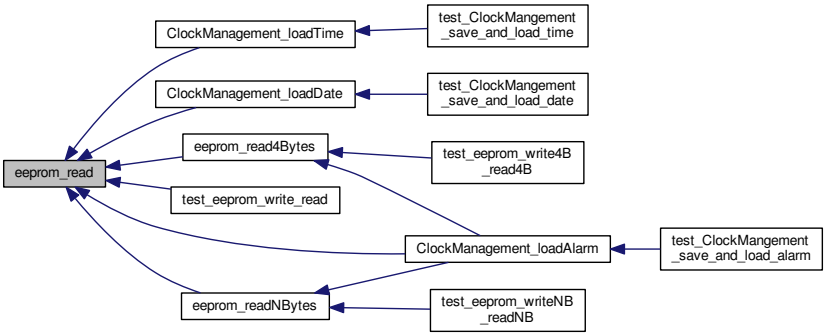
uint8_t	data from eeprom

Definition at line 156 of file NPC_eeprom.c.

Here is the call graph for this function:



Here is the caller graph for this function:



3.27.2.3 uint32_t eeprom_read4Bytes (uint16_t *baseAddress*)

Read a 32byte from eeprom.

Parameters

baseAddress	

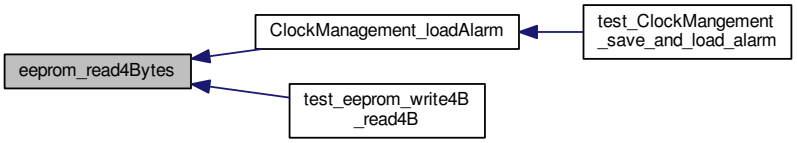
Return values

uint32_t	

Definition at line 295 of file NPC_eeprom.c.

Here is the call graph for this function:

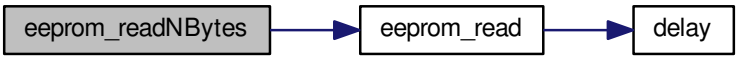
Here is the caller graph for this function:



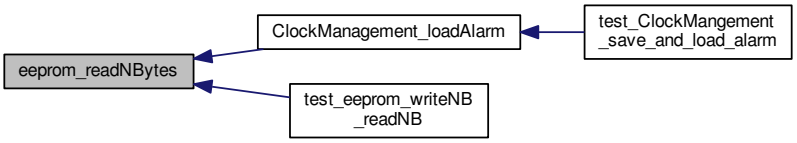
3.27.2.4 void eeprom_readNBytes (uint16_t *baseAddress,* uint8_t * *data,* uint16_t *N*)

Definition at line 309 of file NPC_eeprom.c.

Here is the call graph for this function:



Here is the caller graph for this function:



3.27.2.5 ErrorStatus eeprom_write (uint16_t *address,* uint8_t *data*)

Write a byte to the eeprom.

Parameters

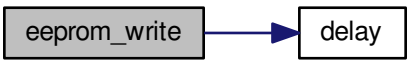
address	The address of th memory
data	The data to be written to the memory

Return values

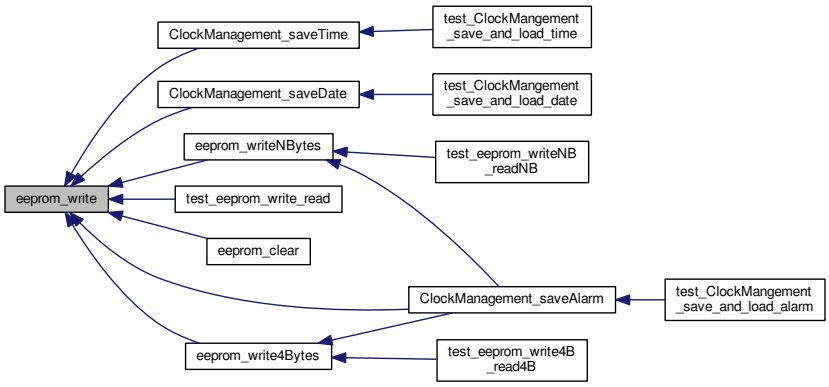
None	

Definition at line 108 of file NPC_eeprom.c.

Here is the call graph for this function:



Here is the caller graph for this function:



3.27.2.6 ErrorStatus eeprom_write32Bytes (uint16_t *baseAddress*, uint8_t * *data*)

Write a page to the eeprom.

Parameters

baseAddress	The base address of the page
data	An array of data to be send

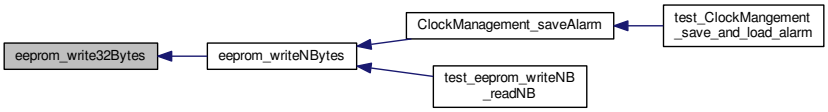
Return values

None	

Definition at line 192 of file NPC_eeprom.c.

Here is the call graph for this function:

Here is the caller graph for this function:



3.27.2.7 ErrorStatus eeprom_write4Bytes (uint16_t *baseAddress*, uint8_t * *data*)

Write 4 bytes to the eeprom.

Parameters

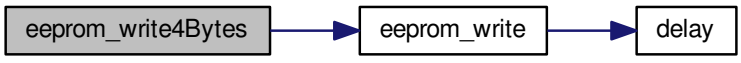
baseAddress	address of data
data	data to be written

Return values

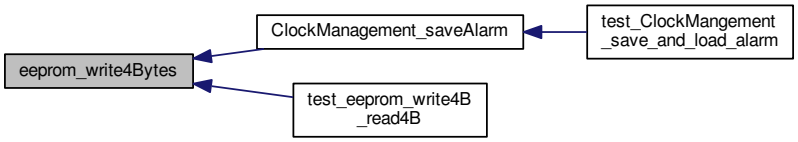
ErrorStatus	

Definition at line 279 of file NPC_eeprom.c.

Here is the call graph for this function:



Here is the caller graph for this function:



3.27.2.8 ErrorStatus eeprom_writeNBytes (uint16_t *baseAddress*, uint8_t * *data*, uint16_t *N*)

Write N bytes to the eeprom.

Note

N is divided into pages before been written to memory

Parameters

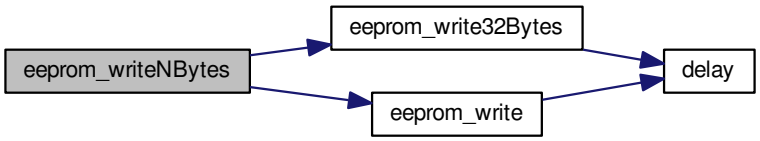
baseAddress	address of data
data	data to be written
N	number of data to write

Return values

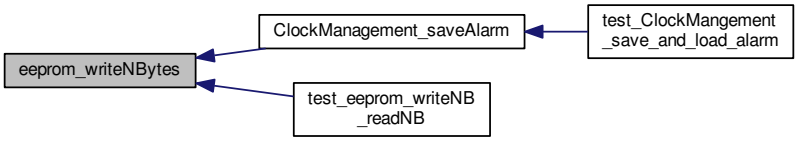
ErrorStatus	

Definition at line 259 of file NPC_eeprom.c.

Here is the call graph for this function:



Here is the caller graph for this function:



3.28 Initialisation functions

Neopixel initialisation functions.

Collaboration diagram for Initialisation functions:

Functions

- void neopixel_init (void)
  Initialise the neopixel.

3.28.1 Detailed Description

Neopixel initialisation functions.

3.28.2 Function Documentation

3.28.2.1 void neopixel_init (void)

Initialise the neopixel.

Parameters

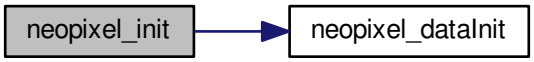
None	

Return values

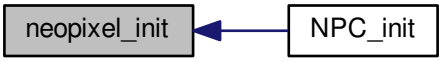
None	

Definition at line 51 of file NPC_neopixel.c.

Here is the call graph for this function:



Here is the caller graph for this function:



3.29 State alteration functions

Neopixel state alteration functions.

Collaboration diagram for State alteration functions:

Functions

- void neopixel_setBrightness (uint8_t b)
  Set the brightness of the led.
- void neopixel_setState (uint8_t s)
- void neopixel_show (void)
- void neopixel_clear (void)
  Stop pushing data to the neopixels.
- void neopixel_dataInit (void)
  Initialise the LEDbuffer.
- void TIM2_IRQHandler (void)
  Timer Handler for neopixel.
- void neopixel_setPixelColourRGB (uint8_t n, uint8_t r, uint8_t g, uint8_t b)
  Set the colour of one led.

3.29.1 Detailed Description

Neopixel state alteration functions.

3.29.2 Function Documentation

3.29.2.1 void neopixel_clear (void)

Stop pushing data to the neopixels.

Parameters

None	

Return values

None	

Definition at line 204 of file NPC_neopixel.c.

Here is the call graph for this function:

3.29.2.2 void neopixel_dataInit (void)

Initialise the LEDbuffer.

Parameters

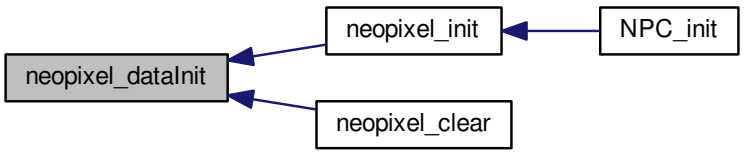
None	

Return values

None	

Definition at line 214 of file NPC_neopixel.c.

Here is the caller graph for this function:



3.29.2.3 void neopixel_setBrightness (uint8_t *b*)

Set the brightness of the led.

Note

- completely dim: 0
  - fully bright: 255

Parameters

b	Brightness

Return values

None	

Definition at line 278 of file NPC_neopixel.c.

3.29.2.4 void neopixel_setPixelColourRGB (uint8_t *n*, uint8_t *r*, uint8_t *g*, uint8_t *b*)

Set the colour of one led.

Parameters

n	Led index
r	RED intensity
g	GREEN intensity
b	BLUE intensity

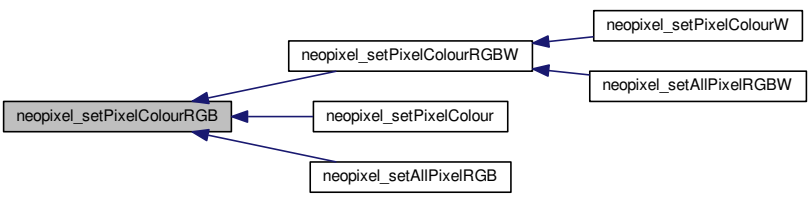
Return values

None	

Definition at line 244 of file NPC_neopixel.c.

Here is the call graph for this function:

Here is the caller graph for this function:



3.29.2.5 void neopixel_setState (uint8_t *s*)

3.29.2.6 void neopixel_show (void)

3.29.2.7 void TIM2_IRQHandler (void)

Timer Handler for neopixel.

Note

LEDBuffer is pushed every time the handle is called

Parameters

None	

Return values

None	

Definition at line 183 of file NPC_neopixel.c.

3.30 Colour generation functions

Neopixel colour functions.

Collaboration diagram for Colour generation functions:

Functions

- uint32_t neopixel_colourRGB (uint8_t r, uint8_t g, uint8_t b)
  convert RGB 3 8bit colour to a 32bit colour
- uint32_t neopixel_colourRGBW (uint8_t r, uint8_t g, uint8_t b, uint8_t w)
  convert RGB 3 8bit colour to a 32bit colour

3.30.1 Detailed Description

Neopixel colour functions.

3.30.2 Function Documentation

3.30.2.1 uint32_t neopixel_colourRGB (uint8_t *r*, uint8_t *g*, uint8_t *b*)

convert RGB 3 8bit colour to a 32bit colour

Note

MS3 0, MS2 r, MS1 g, MS0 b

Parameters

r	RED intensity
g	GREEN intensity
b	BLUE intensity

Return values

None	

Definition at line 298 of file NPC_neopixel.c.

3.30.2.2 uint32_t neopixel_colourRGBW (uint8_t *r,* uint8_t *g,* uint8_t *b,* uint8_t *w*)

convert RGB 3 8bit colour to a 32bit colour

Note

MS3 w, MS2 r, MS1 g, MS0 b

Parameters

r	RED intensity
g	GREEN intensity
b	BLUE intensity
w	WHITE intensity

Return values

None	

Definition at line 311 of file NPC_neopixel.c.

3.31 colour display functions

Neopixel colour display functions.

Collaboration diagram for colour display functions:

Functions

- void neopixel_setPixelColourRGB (uint8_t n, uint8_t r, uint8_t g, uint8_t b)
  Set the colour of one led.
- void neopixel_setPixelColourRGBW (uint8_t n, uint8_t r, uint8_t g, uint8_t b, uint8_t w)
  Set the colour of one led.
- void neopixel_setPixelColour (uint8_t n, uint32_t c)
  Set the colour of one led.
- void neopixel_setPixelColourW (uint8_t n, uint32_t c)
  Set the colour of one led.
- void neopixel_setAllPixelRGB (uint8_t r, uint8_t g, uint8_t b)
  set all the pixel on the line to a specific colour
- void neopixel_setAllPixelRGBW (uint8_t r, uint8_t g, uint8_t b, uint8_t w)
  set all the pixel on the line to a specific colour

3.31.1 Detailed Description

Neopixel colour display functions.

3.31.2 Function Documentation

3.31.2.1 void neopixel_setAllPixelRGB (uint8_t *r*, uint8_t *g*, uint8_t *b*)

set all the pixel on the line to a specific colour

Parameters

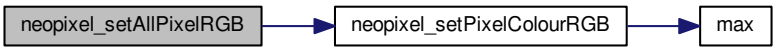
r	RED intensity
g	GREEN intensity
b	BLUE intensity

Return values

None	

Definition at line 374 of file NPC_neopixel.c.

Here is the call graph for this function:



3.31.2.2 void neopixel_setAllPixelRGBW (uint8_t *r*, uint8_t *g*, uint8_t *b*, uint8_t *w*)

set all the pixel on the line to a specific colour

Parameters

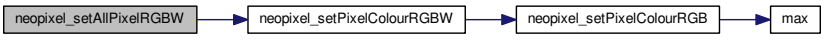
r	RED intensity
g	GREEN intensity
b	BLUE intensity
w	WHITE intensity

Return values

None	

Definition at line 388 of file NPC_neopixel.c.

Here is the call graph for this function:



3.31.2.3 void neopixel_setPixelColour (uint8_t *n*, uint32_t *c*)

Set the colour of one led.

Parameters

n	Led index
c	32bit RGB colour

Return values

None	

Definition at line 353 of file NPC_neopixel.c.

Here is the call graph for this function:

3.31.2.4 void neopixel_setPixelColourRGB (uint8_t *n*, uint8_t *r*, uint8_t *g*, uint8_t *b*)

Set the colour of one led.

Parameters

n	Led index
r	RED intensity
g	GREEN intensity
b	BLUE intensity

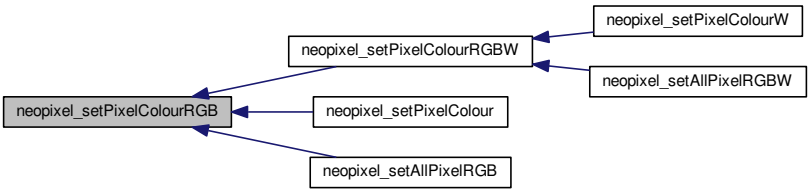
Return values

None	

Definition at line 244 of file NPC_neopixel.c.

Here is the call graph for this function:

Here is the caller graph for this function:



3.31.2.5 void neopixel_setPixelColourRGBW (uint8_t *n*, uint8_t *r*, uint8_t *g*, uint8_t *b*, uint8_t *w*)

Set the colour of one led.

Parameters

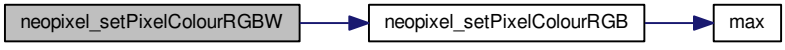
n	Led index
r	RED intensity
g	GREEN intensity
b	BLUE intensity
w	WHITE intensity

Return values

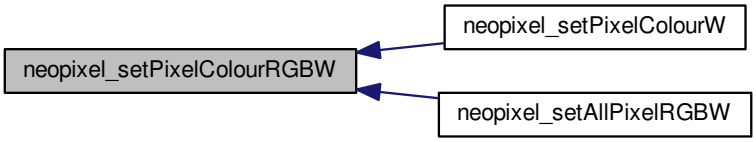
None	

Definition at line 333 of file NPC_neopixel.c.

Here is the call graph for this function:



Here is the caller graph for this function:



3.31.2.6 void neopixel_setPixelColourW (uint8_t *n*, uint32_t *c*)

Set the colour of one led.

Parameters

n	Led index
c	32bit RGB colour

Return values

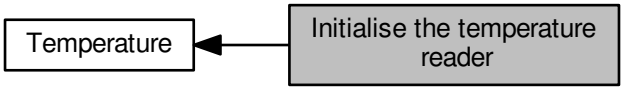
None	

Definition at line 363 of file NPC_neopixel.c.

Here is the call graph for this function:

3.32 Initialise the temperature reader

Collaboration diagram for Initialise the temperature reader:



Functions

- void temperature_init (void)

  Initialise the ADC.

3.32.1 Detailed Description

3.32.2 Function Documentation

3.32.2.1 void temperature_init (void)

Initialise the ADC.

Note

This function configure the ADC peripheral 1) Enable peripheral clocks 3) Configure ADC channel8 pin as analog input 4) Configure ADC1 channel 1

Parameters

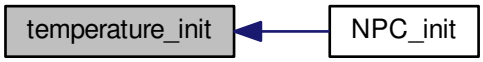
None	

Return values

None	

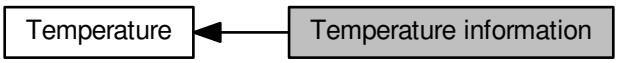
Definition at line 51 of file NPC_temperature.c.

Here is the caller graph for this function:



3.33 Temperature information

Collaboration diagram for Temperature information:



Functions

- uint16_t temperature_value (void)

  Read ADC value.
- int32_t temperature_read (void)

  Convert the ADC value to its corresponding temperature value.

3.33.1 Detailed Description

3.33.2 Function Documentation

3.33.2.1 int32_t temperature_read (void)

Convert the ADC value to its corresponding temperature value.

Parameters

None	

Return values

int32_t	of the temperature read

Definition at line 106 of file NPC_temperature.c.

Here is the call graph for this function:

3.33.2.2 uint16_t temperature_value (void)

Read ADC value.

Parameters

None	

Return values

uint32_t	of the ADC value

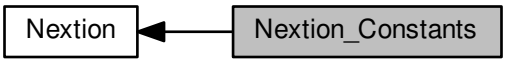
Definition at line 94 of file NPC_temperature.c.

Here is the caller graph for this function:

3.34 Nextion_Constants

define nextion constant

Collaboration diagram for Nextion_Constants:



Macros

- #define NEXTION_PERIPH_USARTX RCC_APB1Periph_USART2
- #define NEXTION_PERIPH_GPIOX RCC_AHB1Periph_GPIOA
- #define NEXTION_GPIOX GPIOA
- #define NEXTION_TX_PIN GPIO_Pin_2
- #define NEXTION_RX_PIN GPIO_Pin_3
- #define NEXTION_TX_PINSOURCE GPIO_PinSource2
- #define NEXTION_RX_PINSOURCE GPIO_PinSource3
- #define NEXTION_AF_USART GPIO_AF_USART2
- #define NEXTION_USARTX USART2
- #define NEXTION_USARTX_IRQ USART2_IRQn
- #define NEXTION_BAUDRATE 9600

3.34.1 Detailed Description

define nextion constant

3.34.2 Macro Definition Documentation

3.34.2.1 #define NEXTION_AF_USART GPIO_AF_USART2

Definition at line 48 of file NPSC_nextion.h.

3.34.2.2 #define NEXTION_BAUDRATE 9600

Definition at line 51 of file NPSC_nextion.h.

3.34.2.3 #define NEXTION_GPIOX GPIOA

Definition at line 43 of file NPSC_nextion.h.

3.34.2.4 #define NEXTION_PERIPH_GPIOX RCC_AHB1Periph_GPIOA

Definition at line 42 of file NPSC_nextion.h.

3.34.2.5 #define NEXTION_PERIPH_USARTX RCC_APB1Periph_USART2

Definition at line 41 of file NPSC_nextion.h.

3.34.2.6 #define NEXTION_RX_PIN GPIO_Pin_3

Definition at line 45 of file NPSC_nextion.h.

3.34.2.7 #define NEXTION_RX_PINSOURCE GPIO_PinSource3

Definition at line 47 of file NPSC_nextion.h.

3.34.2.8 #define NEXTION_TX_PIN GPIO_Pin_2

Definition at line 44 of file NPSC_nextion.h.

3.34.2.9 #define NEXTION_TX_PINSOURCE GPIO_PinSource2

Definition at line 46 of file NPSC_nextion.h.

3.34.2.10 #define NEXTION_USARTX USART2

Definition at line 49 of file NPSC_nextion.h.

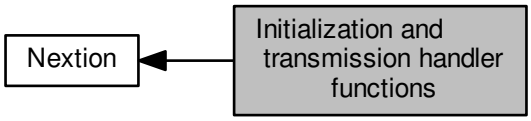
3.34.2.11 #define NEXTION_USARTX_IRQ USART2_IRQn

Definition at line 50 of file NPSC_nextion.h.

3.35 Initialization and transmission handler functions

nextion initialization functions

Collaboration diagram for Initialization and transmission handler functions:



Functions

- void nextion_init (void)

  Initialize the nextion and set baudrate to 9600.
- void USART2_IRQHandler (void)

  Global interrupt handler for USART2.
- void DMA1_Stream5_IRQHandler (void)

  Global interrupt handler for DMA1 stream5.
- void nextion_buffer_update (void)

3.35.1 Detailed Description

nextion initialization functions

3.35.2 Function Documentation

3.35.2.1 void DMA1_Stream5_IRQHandler (void)

Global interrupt handler for DMA1 stream5.

Note

Except memcpy, there is no functions used to

Transfer could be completed by 2 events:

- All data actually transfered (NDTR = 0)
- Stream disabled inside USART IDLE line detected interrupt (NDTR != 0)

Definition at line 156 of file NPSC_nextion.c.

3.35.2.2 void nextion_buffer_update (void)

MUST BE A TASK Loop data back to UART data register

Definition at line 120 of file NPSC_nextion.c.

3.35.2.3 void nextion_init (void)

Initialize the nextion and set baudrate to 9600.

Parameters

None	

Return values

None	

Definition at line 47 of file NPSC_nextion.c.

Here is the caller graph for this function:

3.35.2.4 void USART2_IRQHandler (void)

Global interrupt handler for USART2.

Definition at line 138 of file NPSC_nextion.c.

3.36 Transmission functions

nextion transmission functions

Collaboration diagram for Transmission functions:

Functions

- void nextion_send (uint8_t ∗data)
  send string to the hc-06

3.36.1 Detailed Description

nextion transmission functions

3.36.2 Function Documentation

3.36.2.1 void nextion_send (uint8_t ∗ *data*)

send string to the hc-06

Parameters

data	string to be sent

Return values

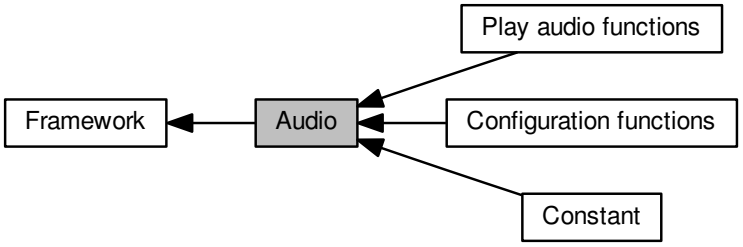
None	

Definition at line 217 of file NPSC_nextion.c.

3.37 Audio

Manage audio configuration and play audio.

Collaboration diagram for Audio:



Modules

- Constant

  Define audio frequency and DMA frequency.
- Configuration functions

  Audio configuration functions.
- Play audio functions

  Audio functions.

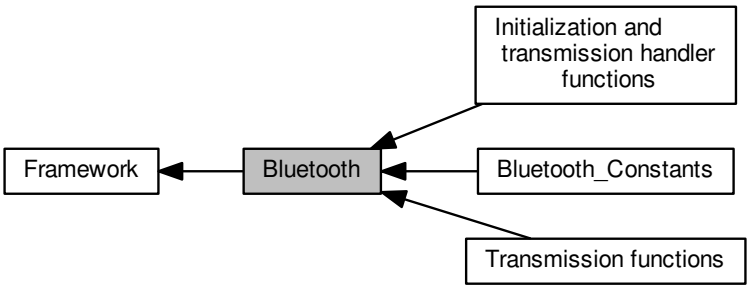
3.37.1 Detailed Description

Manage audio configuration and play audio.

3.38 Bluetooth

bluetooth driver modules

Collaboration diagram for Bluetooth:



Modules

- Bluetooth_Constants
  define bluetooth constant
- Initialization and transmission handler functions
  bluetooth initialization functions
- Transmission functions
  bluetooth transmission functions

Variables

- size_t bluetooth_write
- size_t bluetooth_read

3.38.1 Detailed Description

bluetooth driver modules

3.38.2 Variable Documentation

3.38.2.1 size_t bluetooth_read

Definition at line 56 of file NPC_bluetooth.h.

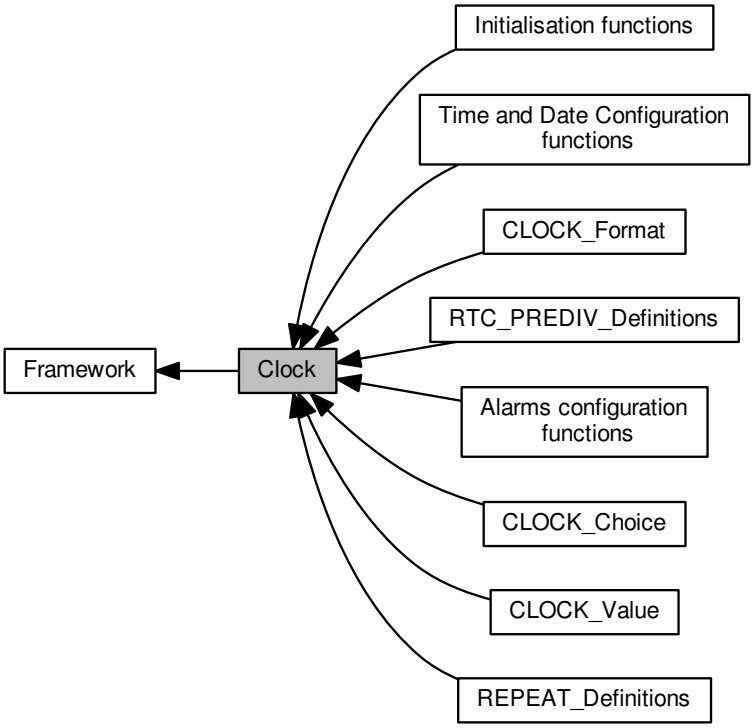
3.38.2.2 size_t bluetooth_write

Definition at line 56 of file NPC_bluetooth.h.

3.39 Clock

Clock driver modules.

Collaboration diagram for Clock:



Modules

- RTC_PREDIV_Definitions
  definition of prescaler for Asynchronous and Synchronous
- CLOCK_Choice
  Clock A or B.
- CLOCK_Format
  AM or PM.
- CLOCK_Value
  Access time or date parameters.
- REPEAT_Definitions
  Alarm repeat options.
- Initialisation functions
  Clock initialisation functions.
- Time and Date Configuration functions
  Clock time and date configuration functions.
- Alarms configuration functions
  Clock alarm configuration functions.

Variables

- RTC_InitTypeDef RTC_InitStruct
- RTC_AlarmTypeDef RTC_AlarmStruct
- EXTI_InitTypeDef EXTI_InitStruct

3.39.1 Detailed Description

Clock driver modules.

3.39.2 Variable Documentation

3.39.2.1 EXTI_InitTypeDef EXTI_InitStruct

Definition at line 60 of file NPC_clock.c.

3.39.2.2 RTC_AlarmTypeDef RTC_AlarmStruct

Definition at line 59 of file NPC_clock.c.

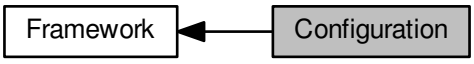
3.39.2.3 RTC_InitTypeDef RTC_InitStruct

Definition at line 58 of file NPC_clock.c.

3.40 Configuration

Configuration driver modules.

Collaboration diagram for Configuration:



Functions

- void NPC_init (void)

  Initialize all firmwares used by the NPC.
- void Error_Handler (void)

  This function is executed in case of error occurrence.

3.40.1 Detailed Description

Configuration driver modules.

3.40.2 Function Documentation

3.40.2.1 void Error_Handler (void)

This function is executed in case of error occurrence.

Parameters

None	

Return values

None	

Definition at line 64 of file NPC_configuration.c.

3.40.2.2 void NPC_init (void)

Initialize all firmwares used by the NPC.

Parameters

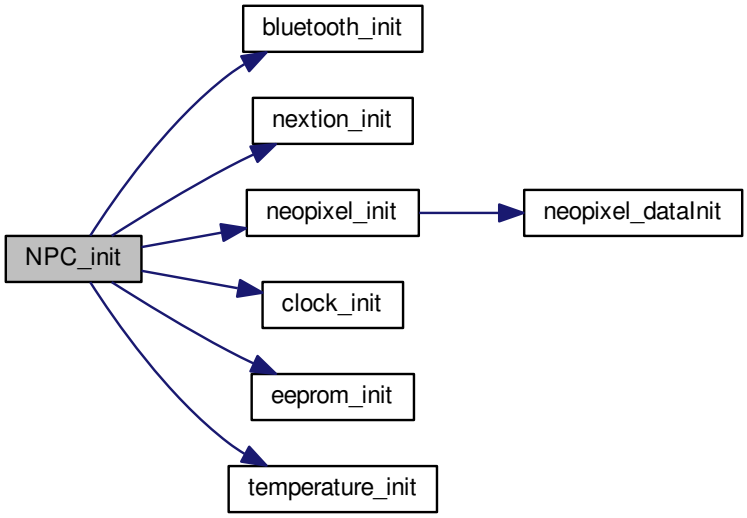
None	

Return values

None	

Definition at line 43 of file NPC_configuration.c.

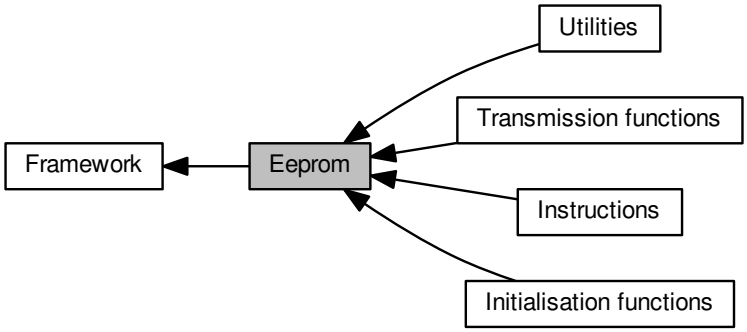
Here is the call graph for this function:



3.41 Eeprom

Eeprom framework.

Collaboration diagram for Eeprom:



Modules

- Instructions

  25LC640A instruction set
- Utilities
- Initialisation functions

  Eeprom initialisation functions.
- Transmission functions

  Eeprom data transmission functions.

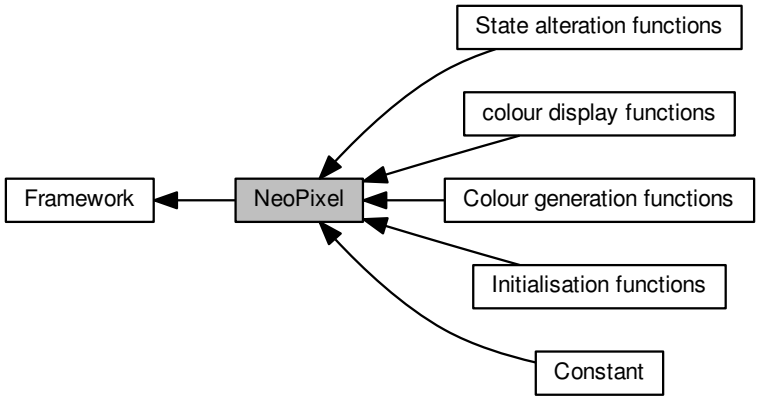
3.41.1 Detailed Description

Eeprom framework.

3.42 NeoPixel

neopixel driver modules

Collaboration diagram for NeoPixel:



Modules

- Constant

  Define audio frequency and DMA frequency.
- Initialisation functions

  Neopixel initialisation functions.
- State alteration functions

  Neopixel state alteration functions.
- Colour generation functions

  Neopixel colour functions.
- colour display functions

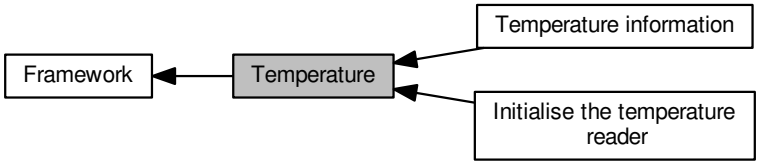
  Neopixel colour display functions.

3.42.1 Detailed Description

neopixel driver modules

3.43 Temperature

Collaboration diagram for Temperature:



Modules

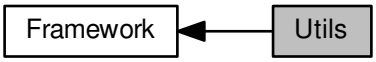
- Initialise the temperature reader
- Temperature information

3.43.1 Detailed Description

3.44 Utils

utils driver modules

Collaboration diagram for Utils:



Macros

- #define UART_BUFFER_SIZE 4
- #define DMA_RX_BUFFER_SIZE 4

Enumerations

- enum bool { false = 0, true = !false }

Functions

- uint32_t max (uint32_t a, uint32_t b, uint32_t c)

  Find max between a, b, and c.
- void delay (uint32_t microseconds)

  delay for the number of microsecond

Variables

- uint8_t pixel_color
- uint8_t DMA_RX_Buffer [DMA_RX_BUFFER_SIZE]
- uint8_t UART_Buffer [UART_BUFFER_SIZE]
- uint8_t pixel_color
- uint8_t DMA_RX_Buffer [DMA_RX_BUFFER_SIZE]
- uint8_t UART_Buffer [UART_BUFFER_SIZE] = {0}

3.44.1 Detailed Description

utils driver modules

3.44.2 Macro Definition Documentation

3.44.2.1 #define DMA_RX_BUFFER_SIZE 4

Definition at line 44 of file NPC_utils.h.

3.44.2.2 #define UART_BUFFER_SIZE 4

Definition at line 43 of file NPC_utils.h.

3.44.3 Enumeration Type Documentation

3.44.3.1 enum bool

Enumerator

false

true

Definition at line 40 of file NPC_utils.h.

3.44.4 Function Documentation

3.44.4.1 void delay (uint32_t *microseconds*)

delay for the number of microsecond

Note

TODO use RTOS delay instead

Parameters

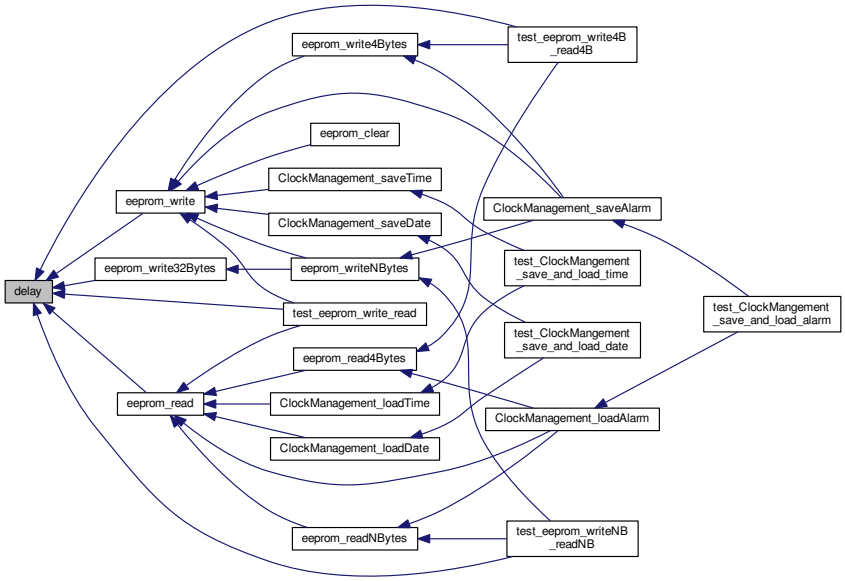
microseconds	

Return values

None	

Definition at line 59 of file NPC_utils.c.

Here is the caller graph for this function:



3.44.4.2 uint32_t max (uint32_t *a*, uint32_t *b*, uint32_t *c*)

Find max between a, b, and c.

Parameters

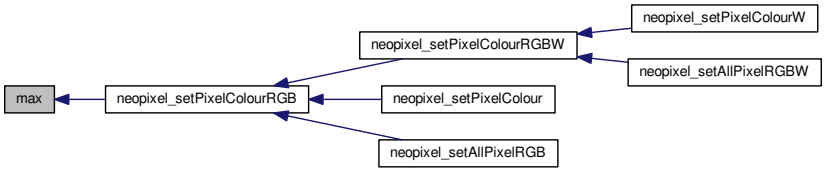
a	First value
b	second value
c	Third value

Return values

uint32_t	of the maximum value

Definition at line 49 of file NPC_utils.c.

Here is the caller graph for this function:



3.44.5 Variable Documentation

3.44.5.1 uint8_t DMA_RX_Buffer[DMA_RX_BUFFER_SIZE]

Definition at line 38 of file NPC_utils.c.

3.44.5.2 uint8_t DMA_RX_Buffer[DMA_RX_BUFFER_SIZE]

Definition at line 38 of file NPC_utils.c.

3.44.5.3 uint8_t pixel_color

Definition at line 36 of file NPC_utils.c.

3.44.5.4 uint8_t pixel_color

Definition at line 36 of file NPC_utils.c.

3.44.5.5 uint8_t UART_Buffer[UART_BUFFER_SIZE] = {0}

Definition at line 39 of file NPC_utils.c.

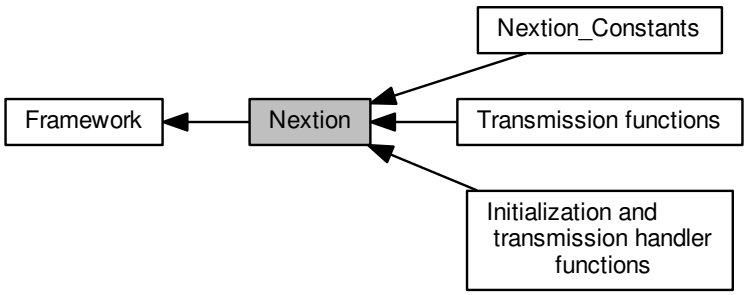
3.44.5.6 uint8_t UART_Buffer[UART_BUFFER_SIZE]

Definition at line 39 of file NPC_utils.c.

3.45 Nextion

nextion driver modules

Collaboration diagram for Nextion:



Modules

- Nextion_Constants
  define nextion constant
- Initialization and transmission handler functions
  nextion initialization functions
- Transmission functions
  nextion transmission functions

Variables

- size_t nextion_write
- size_t nextion_read

3.45.1 Detailed Description

nextion driver modules

3.45.2 Variable Documentation

3.45.2.1 size_t nextion_read

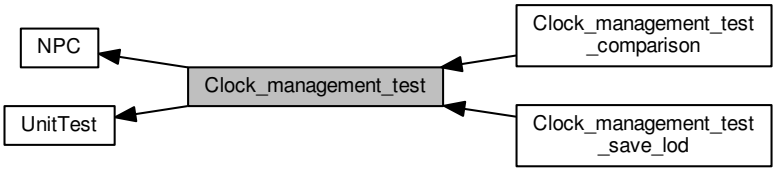
Definition at line 56 of file NPSC_nextion.h.

3.45.2.2 size_t nextion_write

Definition at line 56 of file NPSC_nextion.h.

3.46 Clock_management_test

Collaboration diagram for Clock_management_test:



Modules

- Clock_management_test_save_lod

  Save and Load unit test.
- Clock_management_test_comparison

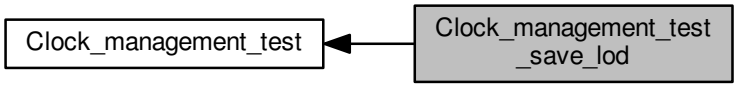
  comparison unit test

3.46.1 Detailed Description

3.47 Clock_management_test_save_lod

Save and Load unit test.

Collaboration diagram for Clock_management_test_save_lod:



Functions

- bool test_ClockMangement_save_and_load_time (void)

  Unit testing for time save and load.
- bool test_ClockMangement_save_and_load_date (void)

  Unit testing for date save and load.
- bool test_ClockMangement_save_and_load_alarm (void)

  Unit testing for alarm save and load.

3.47.1 Detailed Description

Save and Load unit test.

3.47.2 Function Documentation

3.47.2.1 bool test_ClockMangement_save_and_load_alarm (void)

Unit testing for alarm save and load.

Parameters

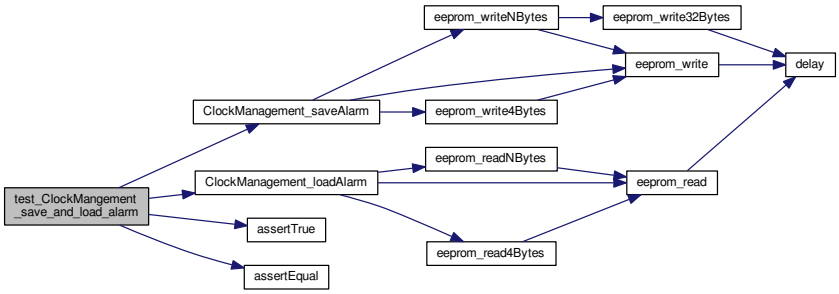
None	

Return values

None	

Definition at line 97 of file clock_management_test.c.

Here is the call graph for this function:



3.47.2.2 bool test_ClockMangement_save_and_load_date (void)

Unit testing for date save and load.

Parameters

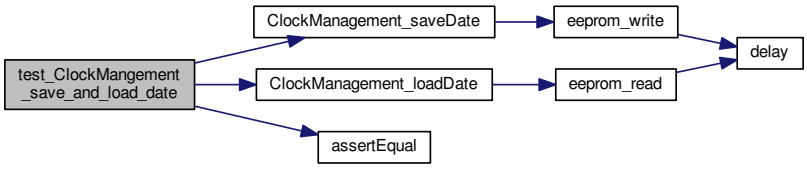
None	

Return values

None	

Definition at line 73 of file clock_management_test.c.

Here is the call graph for this function:



3.47.2.3 bool test_ClockMangement_save_and_load_time (void)

Unit testing for time save and load.

Parameters

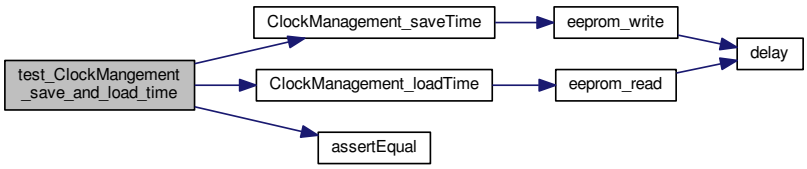
None	

Return values

None	

Definition at line 49 of file clock_management_test.c.

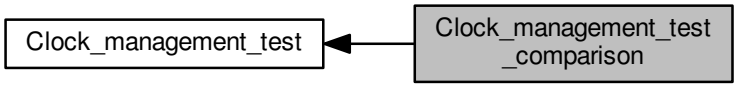
Here is the call graph for this function:



3.48 Clock_management_test_comparison

comparison unit test

Collaboration diagram for Clock_management_test_comparison:



Functions

- bool test_ClockMangement_time_comparison (void)
  Unit testing for time comparison.
- bool test_ClockMangement_date_comparison (void)
  Unit testing for date comparison.
- bool test_ClockMangement_alarm_comparison (void)
  Unit testing for alarm comparison.

3.48.1 Detailed Description

comparison unit test

3.48.2 Function Documentation

3.48.2.1 bool test_ClockMangement_alarm_comparison (void)

Unit testing for alarm comparison.

Parameters

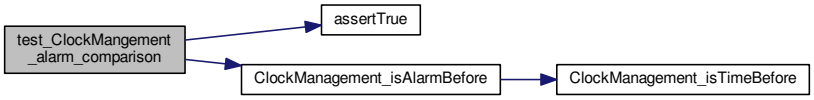
None	

Return values

bool	

Definition at line 222 of file clock_management_test.c.

Here is the call graph for this function:



3.48.2.2 bool test_ClockMangement_date_comparison (void)

Unit testing for date comparison.

Parameters

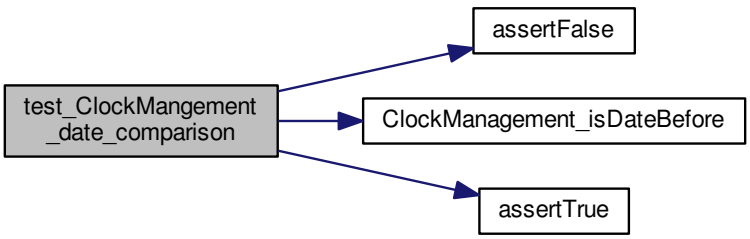
None	

Return values

bool	

Definition at line 181 of file clock_management_test.c.

Here is the call graph for this function:



3.48.2.3 bool test_ClockMangement_time_comparison (void)

Unit testing for time comparison.

Parameters

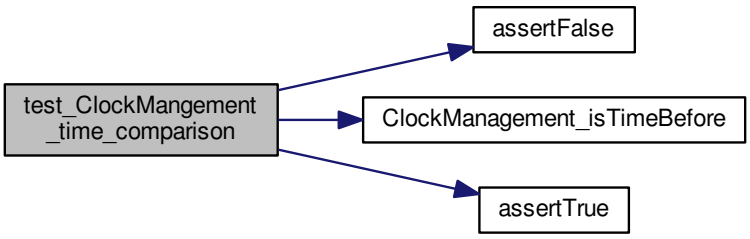
None	

Return values

bool	

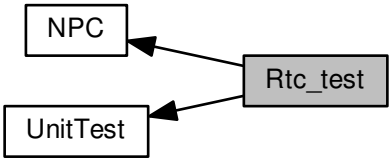
Definition at line 141 of file clock_management_test.c.

Here is the call graph for this function:



3.49 Rtc_test

Collaboration diagram for Rtc_test:



Functions

- bool test_clock_date (void)

  Unit test for date save and load.
- bool test_clock_time (void)

  Unit test for time save and load.
- bool test_clock_alarm (void)

  Unit test for alarm save and load.

3.49.1 Detailed Description

3.49.2 Function Documentation

3.49.2.1 bool test_clock_alarm (void)

Unit test for alarm save and load.

Parameters

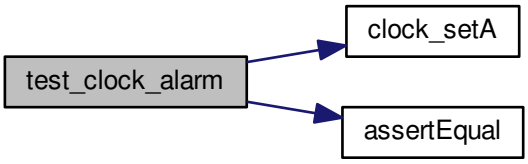
None	

Return values

bool	

Definition at line 84 of file rtc_test.c.

Here is the call graph for this function:



3.49.2.2 bool test_clock_date (void)

Unit test for date save and load.

Parameters

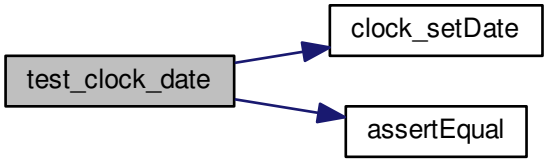
None	

Return values

bool	

Definition at line 42 of file rtc_test.c.

Here is the call graph for this function:



3.49.2.3 bool test_clock_time (void)

Unit test for time save and load.

Parameters

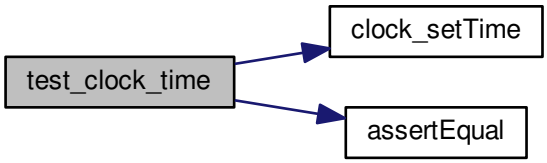
None	

Return values

bool	

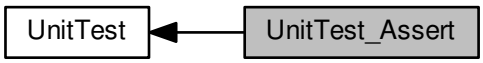
Definition at line 63 of file rtc_test.c.

Here is the call graph for this function:



3.50 UnitTest_Assert

Collaboration diagram for UnitTest_Assert:



Functions

- bool assertTrue (bool condition)
  Assert than condition is true.
- bool assertFalse (bool condition)
  Assert than condition is false.
- bool assertEqual (int a, int b)
  Assert than a equals b.
- bool assertGreater (int a, int b)
  Assert than a is greater than b.
- bool assertLess (int a, int b)
  Assert than a less than than b.
- bool assertGreaterOrEqual (int a, int b)
  Assert than a is greater or equals to b.
- bool assertLessOrEqual (int a, int b)
  Assert than a is less or equals to b.

3.50.1 Detailed Description

3.50.2 Function Documentation

3.50.2.1 bool assertEqual (int *a*, int *b*)

Assert than a equals b.

Parameters

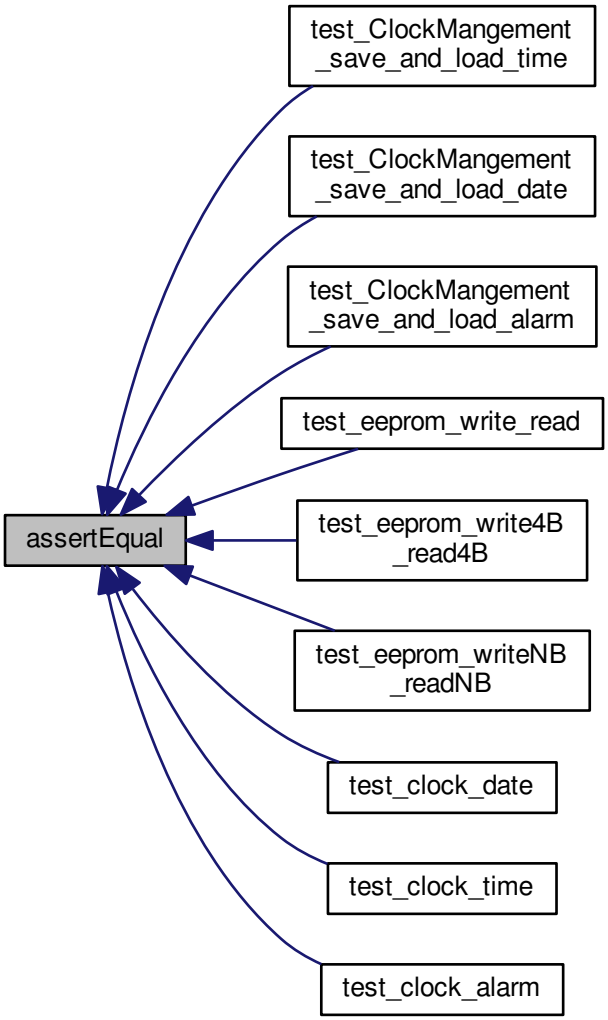
a	
b	

Returns

bool: result of assertion

Definition at line 67 of file unitTest.c.

Here is the caller graph for this function:



3.50.2.2 bool assertFalse (bool *condition*)

Assert than condition is false.

Parameters

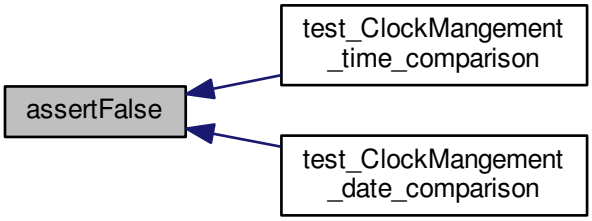
condition	

Returns

bool: result of assertion

Definition at line 57 of file unitTest.c.

Here is the caller graph for this function:



3.50.2.3 bool assertGreater (int *a*, int *b*)

Assert than a is greater than b.

Parameters

a	
b	

Returns

bool: result of assertion

Definition at line 77 of file unitTest.c.

3.50.2.4 bool assertGreaterOrEqual (int *a*, int *b*)

Assert than a is greater or equals to b.

Parameters

a	
b	

Returns

bool: result of assertion

Definition at line 97 of file unitTest.c.

3.50.2.5 bool assertLess (int *a*, int *b*)

Assert than a less than than b.

Parameters

a	
b	

Returns

bool: result of assertion

Definition at line 87 of file unitTest.c.

3.50.2.6 bool assertLessOrEqual (int *a*, int *b*)

Assert than a is less or equals to b.

Parameters

a	
b	

Returns

bool: result of assertion

Definition at line 107 of file unitTest.c.

3.50.2.7 bool assertTrue (bool *condition*)

Assert than condition is true.

Parameters

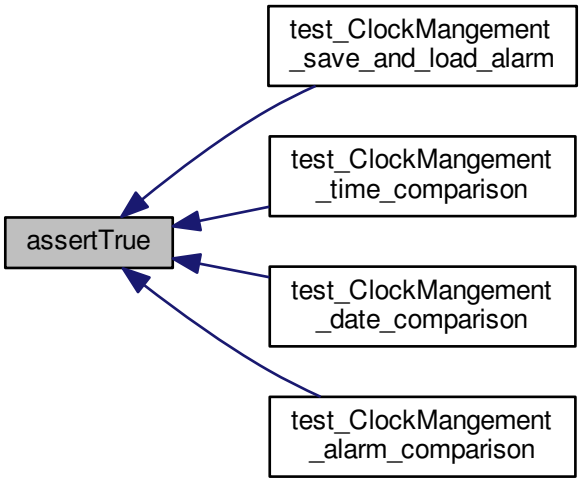
condition	

Returns

bool: result of assertion

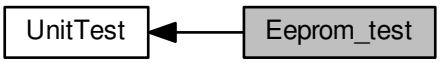
Definition at line 48 of file unitTest.c.

Here is the caller graph for this function:



3.51 Eeprom_test

Collaboration diagram for Eeprom_test:



Functions

- bool test_eeprom_write_read (void)

  Unit test for writing and reading to/from the eeprom.
- bool test_eeprom_write4B_read4B (void)

  Unit test for writing and reading a 32bit number.
- bool test_eeprom_writeNB_readNB (void)

  Unit test for writing and reading a Nbit number.

3.51.1 Detailed Description

3.51.2 Function Documentation

3.51.2.1 bool test_eeprom_write4B_read4B (void)

Unit test for writing and reading a 32bit number.

Parameters

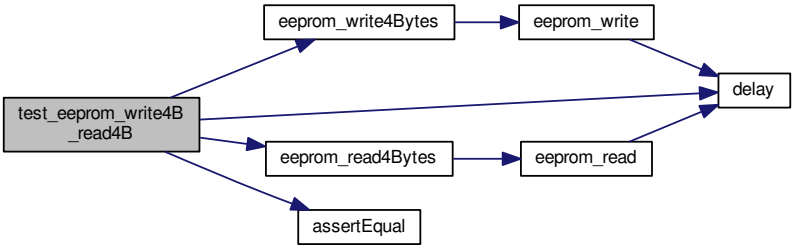
None	

Return values

bool	

Definition at line 61 of file eeprom_test.c.

Here is the call graph for this function:



3.51.2.2 bool test_eeprom_write_read (void)

Unit test for writing and reading to/from the eeprom.

Parameters

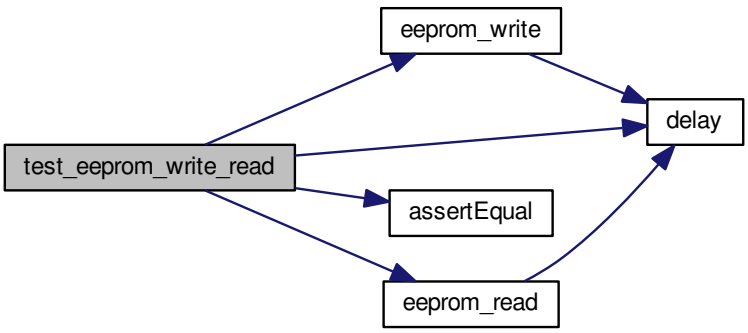
None	

Return values

bool	

Definition at line 44 of file eeprom_test.c.

Here is the call graph for this function:



3.51.2.3 bool test_eeprom_writeNB_readNB (void)

Unit test for writing and reading a Nbit number.

Parameters

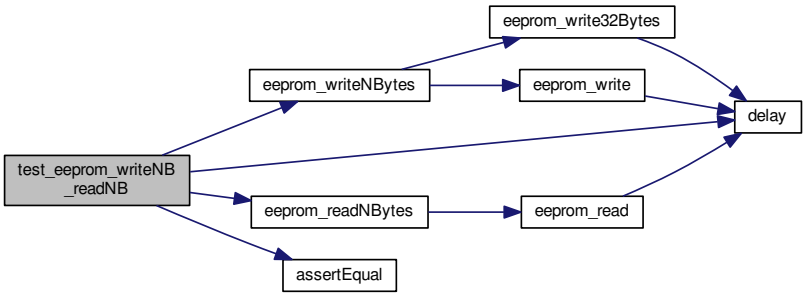
None	

Return values

bool	

Definition at line 76 of file eeprom_test.c.

Here is the call graph for this function:



3.52 UnitTest

Collaboration diagram for UnitTest:

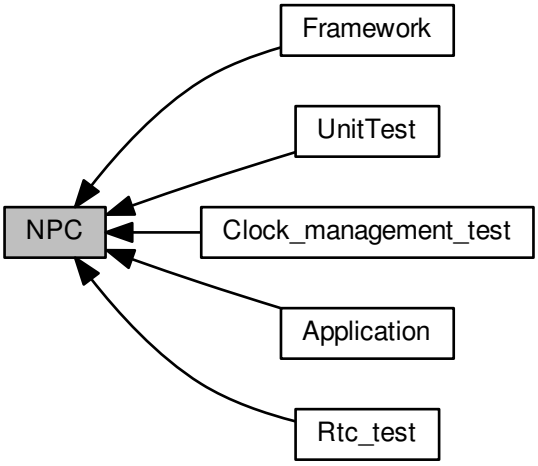
Modules

- UnitTest_Assert

3.52.1 Detailed Description

3.53 NPC

Collaboration diagram for NPC:



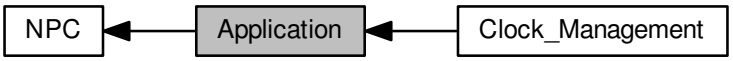
Modules

- Clock_management_test
- Rtc_test
- Application
- Framework
- UnitTest

3.53.1 Detailed Description

3.54 Application

Collaboration diagram for Application:



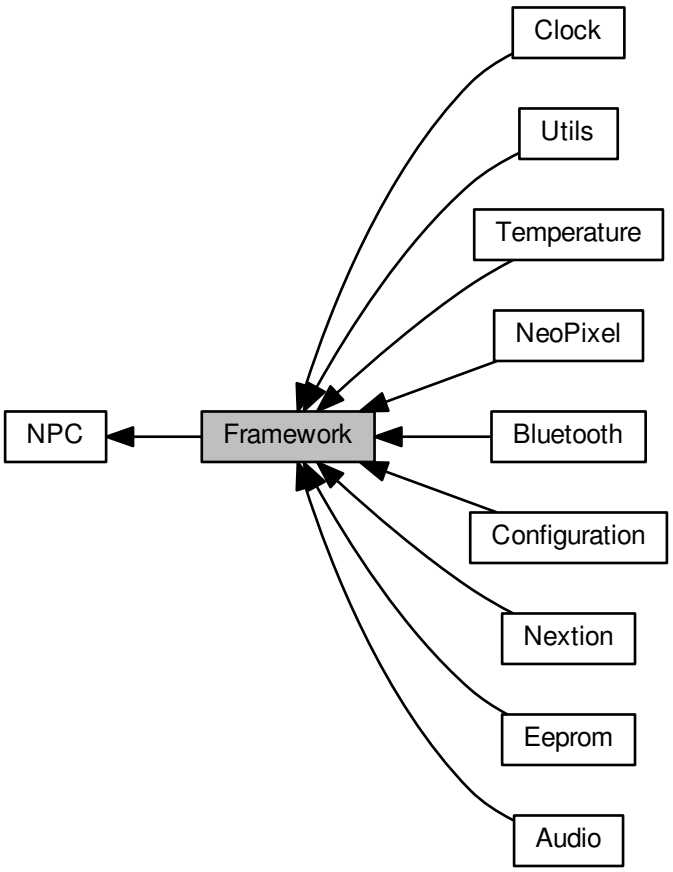
Modules

- Clock_Management

3.54.1 Detailed Description

3.55 Framework

Collaboration diagram for Framework:



Modules

- Audio

  Manage audio configuration and play audio.
- Bluetooth

  bluetooth driver modules
- Clock

  Clock driver modules.
- Configuration

  Configuration driver modules.
- Eeprom

  Eeprom framework.
- NeoPixel

  neopixel driver modules

- Temperature
- Utils

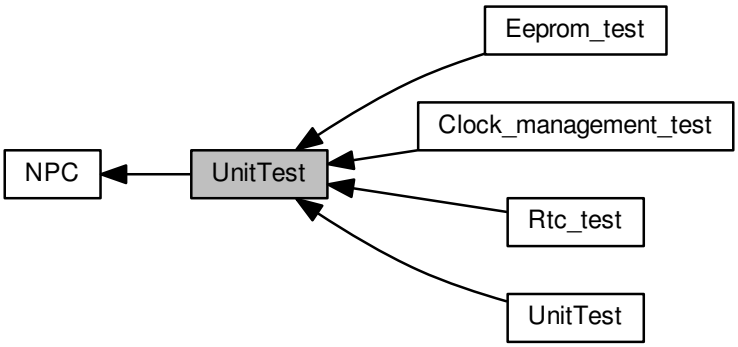
  utils driver modules
- Nextion

  nextion driver modules

3.55.1 Detailed Description

3.56 UnitTest

Collaboration diagram for UnitTest:



Modules

- Clock_management_test
- Eeprom_test
- Rtc_test
- UnitTest

3.56.1 Detailed Description

@

Chapter 4

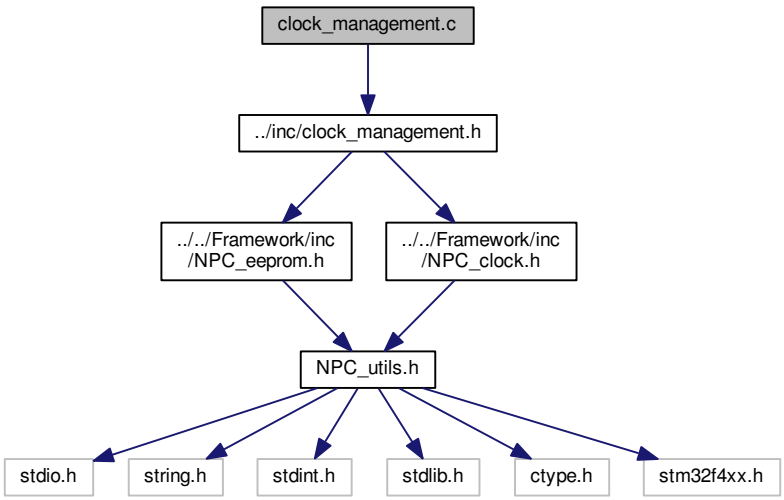
File Documentation

4.1 clock_management.c File Reference

comment

```
#include "../inc/clock_management.h"
```

Include dependency graph for clock_management.c:



Functions

- ErrorStatus ClockManagement_saveAlarm (Alarm_Definition *Alarm_Def, uint16_t address)

  Save an alarm settings to eeprom.
- ErrorStatus ClockManagement_saveTime (RTC_TimeTypeDef *Time_Def)

  Save the time settings to eeprom.
- ErrorStatus ClockManagement_saveDate (RTC_DateTypeDef *Date_Def)

  Save the date settings to eeprom.

- Alarm_Definition ClockManagement_loadAlarm (uint16_t index)

  load an alarm settings from eeprom
- RTC_TimeTypeDef ClockManagement_loadTime (void)

  Load the time settings from eeprom.
- RTC_DateTypeDef ClockManagement_loadDate (void)

  Load the date settings from eeprom.
- bool ClockManagement_isTimeBefore (RTC_TimeTypeDef *time1, RTC_TimeTypeDef *time2)

  Compare two time.
- bool ClockManagement_isDateBefore (RTC_DateTypeDef *date1, RTC_DateTypeDef *date2)

  Convert an date to an integer.
- bool ClockManagement_isAlarmBefore (Alarm_Definition *alarm1, Alarm_Definition *alarm2)

  Compare two alarm.
- void CLockManagement_updateAlarm (void)

  Update the alarm with the closest alarm.

4.1.1 Detailed Description

comment

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

19-March-2017

Attention

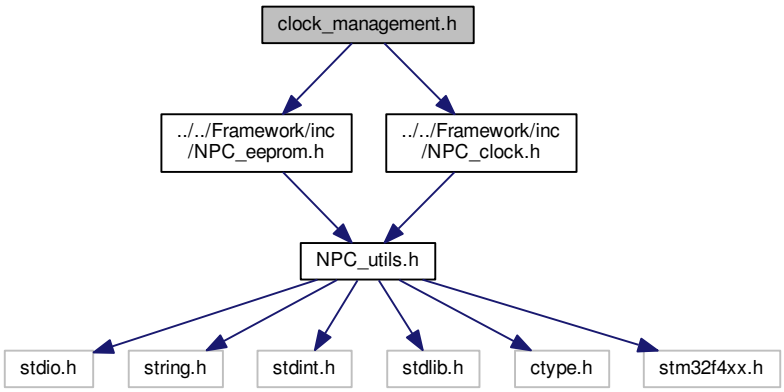
© COPYRIGHT

4.2 clock_management.h File Reference

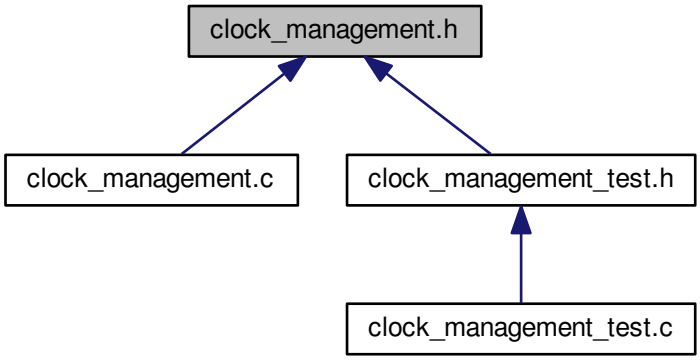
This file contains all the application level functions usable by the user.

```
#include "../../Framework/inc/NPC_eeprom.h"
#include "../../Framework/inc/NPC_clock.h"
```

Include dependency graph for clock_management.h:



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



Data Structures

- struct Alarm_Definition

Macros

- #define TIME_BASE_ADDRESS 0x00
- #define DATE_BASE_ADDRESS 0x04
- #define ALARM_BASE_ADDRESS 0x08
- #define OFFSET_NAME 0x00

- #define OFFSET_DATEWEEKDAY NAME_SIZE
- #define OFFSET_DATEWEEKDAY_SEL OFFSET_DATEWEEKDAY + 1
- #define OFFSET_MASK OFFSET_DATEWEEKDAY_SEL + 4
- #define OFFSET_H12 OFFSET_MASK + 4
- #define OFFSET_HOURS OFFSET_H12 + 1
- #define OFFSET_MINUTES OFFSET_HOURS + 1
- #define OFFSET_SECONDS OFFSET_MINUTES + 1
- #define NAME_SIZE 31

Functions

- ErrorStatus ClockManagement_saveAlarm (Alarm_Definition *Alarm_Def, uint16_t address)
  Save an alarm settings to eeprom.
- ErrorStatus ClockManagement_saveTime (RTC_TimeTypeDef *Time_Def)
  Save the time settings to eeprom.
- ErrorStatus ClockManagement_saveDate (RTC_DateTypeDef *Date_Def)
  Save the date settings to eeprom.
- Alarm_Definition ClockManagement_loadAlarm (uint16_t index)
  load an alarm settings from eeprom
- RTC_TimeTypeDef ClockManagement_loadTime (void)
  Load the time settings from eeprom.
- RTC_DateTypeDef ClockManagement_loadDate (void)
  Load the date settings from eeprom.
- bool ClockManagement_isTimeBefore (RTC_TimeTypeDef *time1, RTC_TimeTypeDef *time2)
  Compare two time.
- bool ClockManagement_isDateBefore (RTC_DateTypeDef *date1, RTC_DateTypeDef *date2)
  Convert an date to an integer.
- bool ClockManagement_isAlarmBefore (Alarm_Definition *alarm1, Alarm_Definition *alarm2)
  Compare two alarm.
- void CLockManagement_updateAlarm (void)
  Update the alarm with the closest alarm.

Variables

- uint16_t eeprom_index
- uint16_t next_alarm

4.2.1 Detailed Description

This file contains all the application level functions usable by the user.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

19-March-2017

Attention

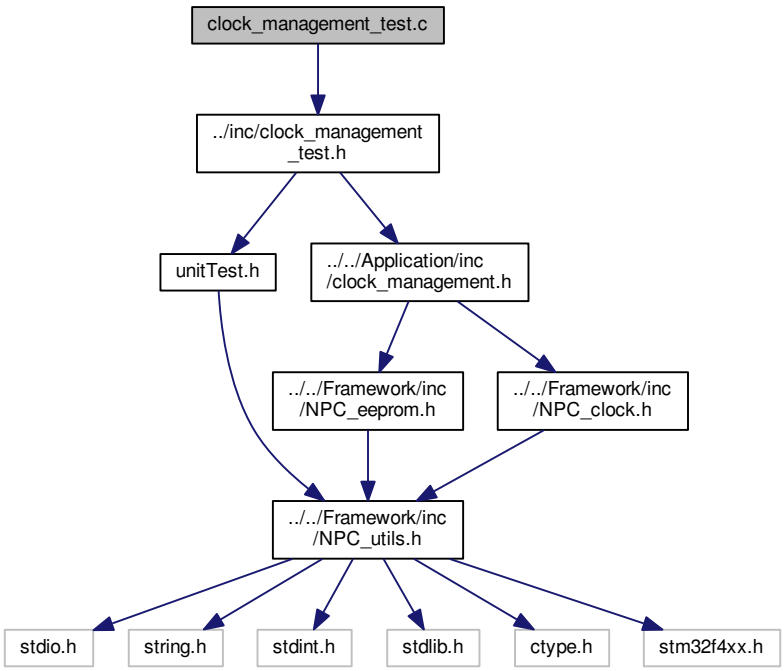
© COPYRIGHT

4.3 clock_management_test.c File Reference

comment

```
#include "../inc/clock_management_test.h"
```

Include dependency graph for clock_management_test.c:



Functions

- bool test_ClockMangement_save_and_load_time (void)

  Unit testing for time save and load.
- bool test_ClockMangement_save_and_load_date (void)

  Unit testing for date save and load.
- bool test_ClockMangement_save_and_load_alarm (void)

  Unit testing for alarm save and load.
- bool test_ClockMangement_time_comparison (void)

  Unit testing for time comparison.
- bool test_ClockMangement_date_comparison (void)

  Unit testing for date comparison.
- bool test_ClockMangement_alarm_comparison (void)

  Unit testing for alarm comparison.

4.3.1 Detailed Description

comment

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

24-March-2017

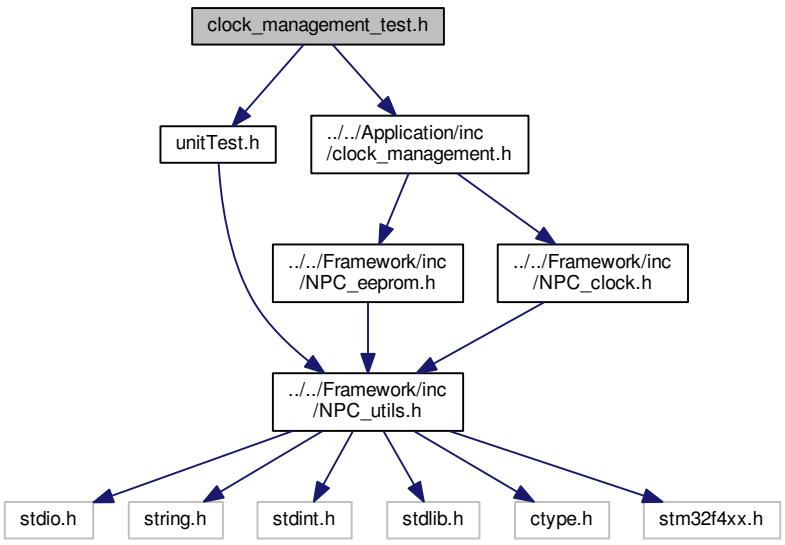
Attention

© COPYRIGHT

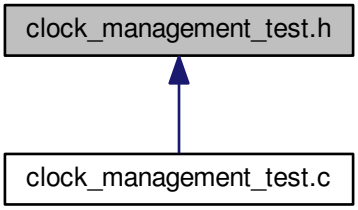
4.4 clock_management_test.h File Reference

```
#include "unitTest.h"
#include "../../Application/inc/clock_management.h"
```

Include dependency graph for clock_management_test.h:



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



Functions

- bool test_ClockMangement_save_and_load_time (void)

  Unit testing for time save and load.
- bool test_ClockMangement_save_and_load_date (void)

  Unit testing for date save and load.
- bool test_ClockMangement_save_and_load_alarm (void)

  Unit testing for alarm save and load.
- bool test_ClockMangement_time_comparison (void)

  Unit testing for time comparison.
- bool test_ClockMangement_date_comparison (void)

  Unit testing for date comparison.
- bool test_ClockMangement_alarm_comparison (void)

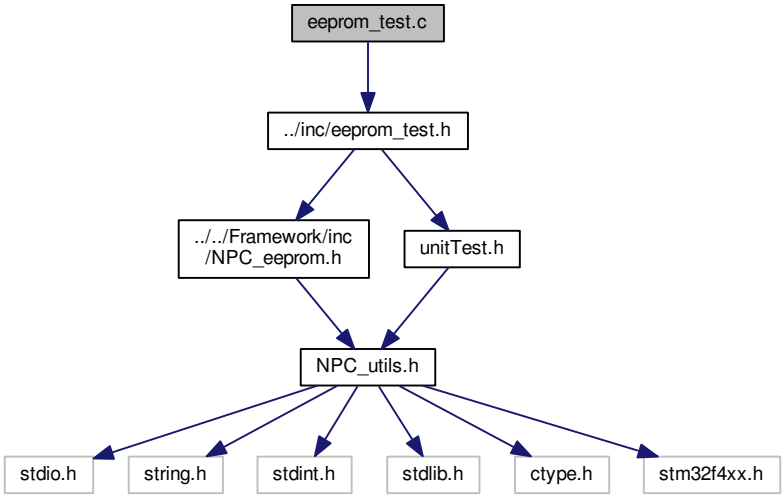
  Unit testing for alarm comparison.

4.5 eeprom_test.c File Reference

This file contains the unit test implementation for the eeprom.

```
#include "../inc/eeprom_test.h"
```

Include dependency graph for eeprom_test.c:



Functions

- bool test_eeprom_write_read (void)

  Unit test for writing and reading to/from the eeprom.
- bool test_eeprom_write4B_read4B (void)

  Unit test for writing and reading a 32bit number.
- bool test_eeprom_writeNB_readNB (void)

  Unit test for writing and reading a Nbit number.

4.5.1 Detailed Description

This file contains the unit test implementation for the eeprom.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

23-March-2017

Attention

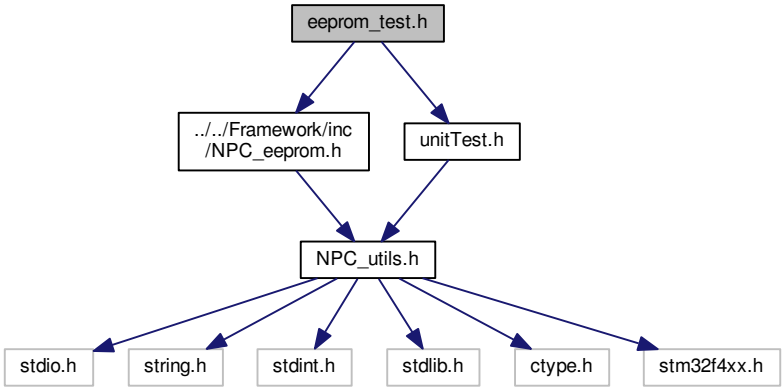
© COPYRIGHT

4.6 eeprom_test.h File Reference

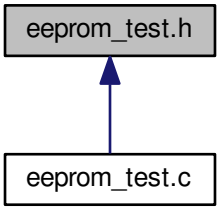
This file contains template of unit tests for the eeprom.

```
#include "../../Framework/inc/NPC_eeprom.h"
#include "unitTest.h"
```

Include dependency graph for eeprom_test.h:



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



Functions

- bool test_eeprom_write_read (void)

  Unit test for writing and reading to/from the eeprom.
- bool test_eeprom_write4B_read4B (void)

  Unit test for writing and reading a 32bit number.
- bool test_eeprom_writeNB_readNB (void)

  Unit test for writing and reading a Nbit number.

4.6.1 Detailed Description

This file contains template of unit tests for the eeprom.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

23-March-2017

Attention

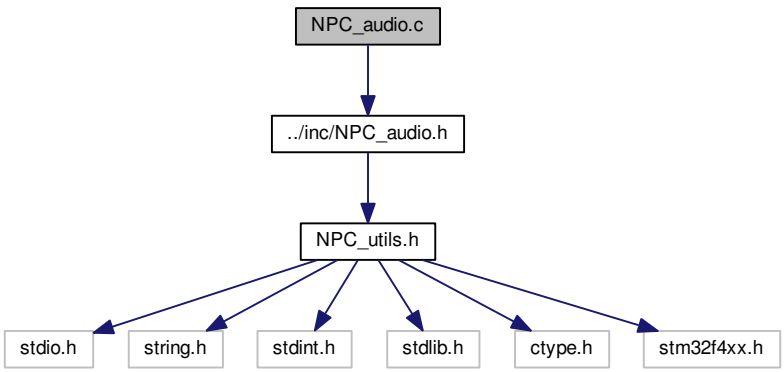
© COPYRIGHT

4.7 NPC_audio.c File Reference

This file provides firmware functions to manage the audio.

```
#include "../inc/NPC_audio.h"
```

Include dependency graph for NPC_audio.c:



Functions

- void audio_disable (void)
  Disable the DMA.
- void audio_init (uint16_t *DACBuffer, uint16_t Size)
  Perform audio initialization.
- void audio_play (uint16_t *DACBuffer, uint16_t Size)
  Play a sample.

4.7.1 Detailed Description

This file provides firmware functions to manage the audio.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

12-March-2017

Attention

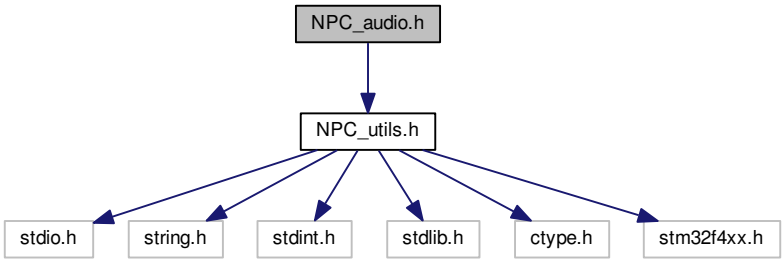
© COPYRIGHT

4.8 NPC_audio.h File Reference

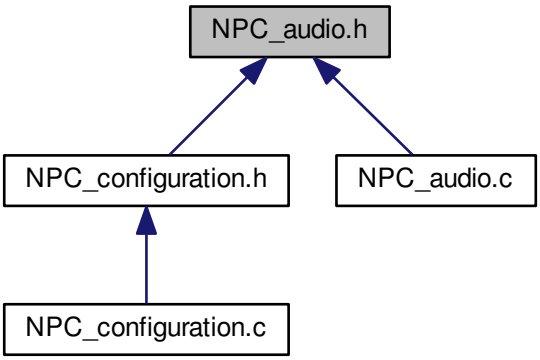
This file contains all the configuration prototypes used by the audio firmware.

```
#include "NPC_utils.h"
```

Include dependency graph for NPC_audio.h:



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



Macros

- #define AUDIO_FREQUENCY 11000
- #define DMA_FREQUENCY (86000000/(2∗AUDIO_FREQUENCY))

Functions

- void audio_disable (void)

  Disable the DMA.
- void audio_init (uint16_t ∗DACBuffer, uint16_t Size)

  Perform audio initialization.
- void audio_play (uint16_t ∗DACBuffer, uint16_t Size)

  Play a sample.

4.8.1 Detailed Description

This file contains all the configuration prototypes used by the audio firmware.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

12-March-2017

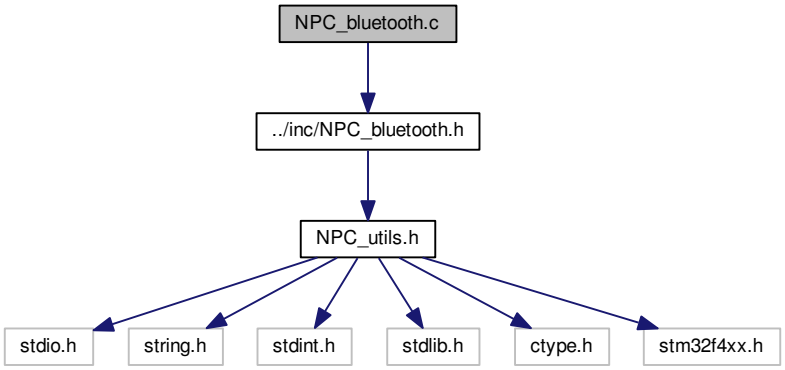
Attention

© COPYRIGHT

4.9 NPC_bluetooth.c File Reference

```
#include "../inc/NPC_bluetooth.h"
```

Include dependency graph for NPC_bluetooth.c:



Functions

- void bluetooth_init (void)

  Initialize the bluetooth and set baudrate to 9600.
- void bluetooth_buffer_update (void)
- void USART1_IRQHandler (void)

  Global interrupt handler for USART1.
- void DMA2_Stream5_IRQHandler (void)

  Global interrupt handler for DMA2 stream5.
- void bluetooth_send (uint8_t *data)

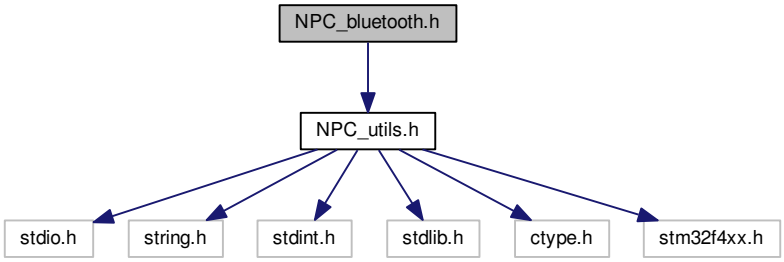
  send string to the hc-06

4.10 NPC_bluetooth.h File Reference

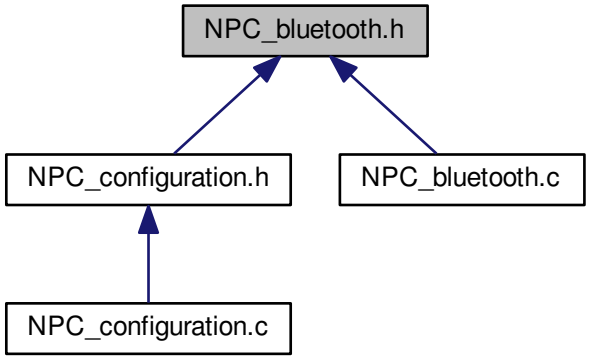
This file contains all the configuration prototypes used by the bluetooth firmware.

```
#include "NPC_utils.h"
```

Include dependency graph for NPC_bluetooth.h:



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



Macros

- #define BLUETOOTH_PERIPH_USARTX RCC_APB2Periph_USART1
- #define BLUETOOTH_PERIPH_GPIOX RCC_AHB1Periph_GPIOB
- #define BLUETOOTH_GPIOX GPIOB
- #define BLUETOOTH_TX_PIN GPIO_Pin_6
- #define BLUETOOTH_RX_PIN GPIO_Pin_7
- #define BLUETOOTH_TX_PINSOURCE GPIO_PinSource6
- #define BLUETOOTH_RX_PINSOURCE GPIO_PinSource7
- #define BLUETOOTH_AF_USART GPIO_AF_USART1
- #define BLUETOOTH_USARTX USART1
- #define BLUETOOTH_USARTX_IRQ USART1_IRQn
- #define BLUETOOTH_BAUDRATE 9600

Functions

- void bluetooth_init (void)

  Initialize the bluetooth and set baudrate to 9600.
- void USART1_IRQHandler (void)

  Global interrupt handler for USART1.
- void DMA2_Stream5_IRQHandler (void)

  Global interrupt handler for DMA2 stream5.
- void bluetooth_send (uint8_t ∗data)

  send string to the hc-06

Variables

- size_t bluetooth_write
- size_t bluetooth_read

4.10.1 Detailed Description

This file contains all the configuration prototypes used by the bluetooth firmware.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

01-March-2017

Attention

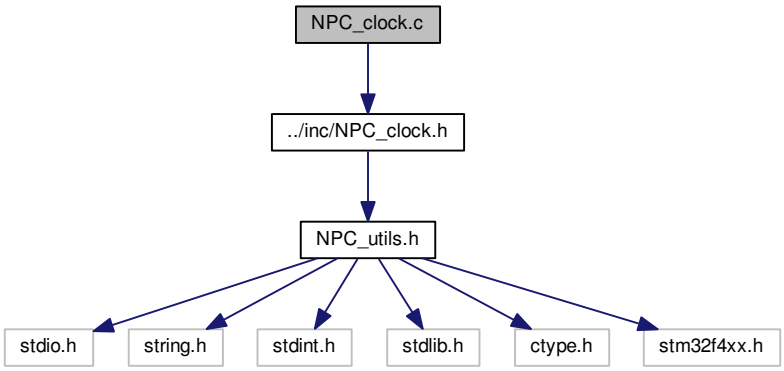
© COPYRIGHT

4.11 NPC_clock.c File Reference

This file provides firmware functions to manage the Date, Time and Alarm of the NPC clock.

```
#include "../inc/NPC_clock.h"
```

Include dependency graph for NPC_clock.c:



Functions

- void clock_init (void)

  Initialise the clock to 1Hz and setup peripherals for Alarm.
- ErrorStatus clock_setDate (uint8_t weekDay, uint8_t month, uint8_t date, uint8_t year)

  Set the clock's date.
- ErrorStatus clock_setTime (uint8_t am_pm, uint8_t hours, uint8_t minutes, uint8_t second)

  Set the clock's time.
- uint32_t clock_getDate (void)

  Get the date encoded in a 32b format.
- uint32_t clock_getTime (void)

  Get the time encoded in a 32b format.
- RTC_AlarmTypeDef clock_createAlarm (uint8_t am_pm, uint8_t hours, uint8_t minutes, uint8_t seconds, uint32_t dateWeekDaySel, uint8_t dateWeekDay, uint32_t repeat)

  Create an Alarm Structure given all the parameters.
- void clock_setA (RTC_AlarmTypeDef *Alarm)

  Set an alarm to RTC_Alarm_A, given a Alarm structure RTC_AlarmTypeDef.
- void clock_setAlarm (uint8_t am_pm, uint8_t hours, uint8_t minutes, uint8_t seconds, uint32_t dateWeek↩ DaySel, uint8_t dateWeekDay, uint32_t repeat)

  Set an alarm to RTC_Alarm_A, given all the alarm parameters.
- void RTC_Alarm_IRQHandler (void)

  Alarm Handler.

Variables

- RTC_InitTypeDef RTC_InitStruct
- RTC_AlarmTypeDef RTC_AlarmStruct
- EXTI_InitTypeDef EXTI_InitStruct

4.11.1 Detailed Description

This file provides firmware functions to manage the Date, Time and Alarm of the NPC clock.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

06-March-2017

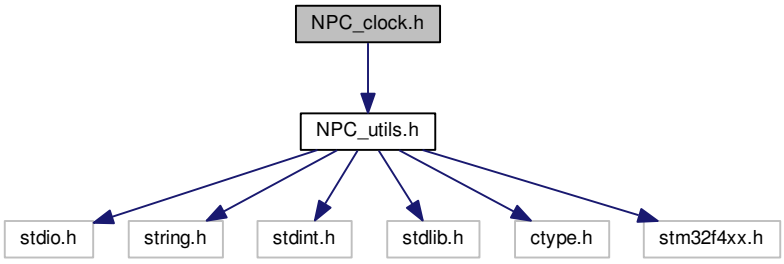
```
 ==============================================================================
                  ##### Note on VBat and VDD #####
 ==============================================================================
 [..]
    (+) Make sure that VBAT is connected to an external power supply
        otherwise RTC info will be lost on VDD power off.
        Ideally RTC info should be saved and loaded from an EEPROM or
        SD card to avoid lost

                  ##### How to use Clock Driver #####
 ==============================================================================
 [..]
   (+) Initialize the Clock using clock_init()
   (+) Set the Date using clock_setDate(...)
   (+) Set the Time using clock_setTime(...)
   (+) Set the Alarm using clock_setAlarm(...)


@attention

<h2><center>&copy; COPYRIGHT </center></h2>
```

4.12 NPC_clock.h File Reference

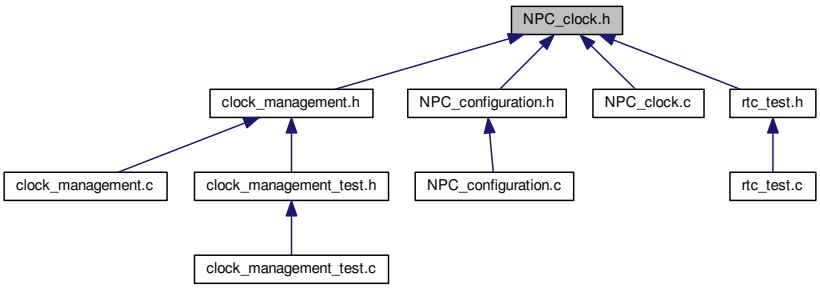
This file contains all the functions prototypes for the clock firmware library used for the NPC.

```
#include "NPC_utils.h"
```

Include dependency graph for NPC_clock.h:



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



Macros

- #define RTC_PREDIV_A 0x7C
- #define RTC_PREDIV_S 0X1F3F
- #define CLOCK_A RTC_Alarm_A
- #define CLOCK_B RTC_Alarm_B
- #define AM RTC_H12_AM
- #define PM RTC_H12_PM
- #define CLOCK_WeekDay (uint8_t) (clock_getDate() >> 24)
- #define CLOCK_Date (uint8_t) (clock_getDate() >> 16)
- #define CLOCK_Month (uint8_t) (clock_getDate() >> 8)
- #define CLOCK_Year (uint8_t) clock_getDate()
- #define CLOCK_Hours (uint8_t) (clock_getTime() >> 24)
- #define CLOCK_Minutes (uint8_t) (clock_getTime() >> 16)
- #define CLOCK_Seconds (uint8_t) (clock_getTime() >> 8)
- #define CLOCK_Format (uint8_t) clock_getTime()
- #define REPEAT_DateWeekDay (uint32_t) RTC_AlarmMask_DateWeekDay
- #define REPEAT_Hours (uint32_t) RTC_AlarmMask_Hours
- #define REPEAT_Minutes (uint32_t) RTC_AlarmMask_Minutes
- #define REPEAT_Seconds (uint32_t) RTC_AlarmMask_Seconds
- #define REPEAT_None (uint32_t) RTC_AlarmMask_None

Functions

- void clock_init (void)

  Initialise the clock to 1Hz and setup peripherals for Alarm.
- ErrorStatus clock_setDate (uint8_t weekDay, uint8_t month, uint8_t date, uint8_t year)

  Set the clock's date.
- ErrorStatus clock_setTime (uint8_t am_pm, uint8_t hours, uint8_t minutes, uint8_t second)

  Set the clock's time.
- uint32_t clock_getDate (void)

  Get the date encoded in a 32b format.
- uint32_t clock_getTime (void)

  Get the time encoded in a 32b format.
- RTC_AlarmTypeDef clock_createAlarm (uint8_t am_pm, uint8_t hours, uint8_t minutes, uint8_t seconds, uint32_t dateWeekDaySel, uint8_t dateWeekDay, uint32_t repeat)

Create an Alarm Structure given all the parameters.

- void clock_setA (RTC_AlarmTypeDef *Alarm)

  Set an alarm to RTC_Alarm_A, given a Alarm structure RTC_AlarmTypeDef.
- void clock_setAlarm (uint8_t am_pm, uint8_t hours, uint8_t minutes, uint8_t seconds, uint32_t dateWeekDaySel, uint8_t dateWeekDay, uint32_t repeat)

  Set an alarm to RTC_Alarm_A, given all the alarm parameters.
- void RTC_Alarm_IRQHandler (void)

  Alarm Handler.

4.12.1 Detailed Description

This file contains all the functions prototypes for the clock firmware library used for the NPC.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

06-March-2017

Attention

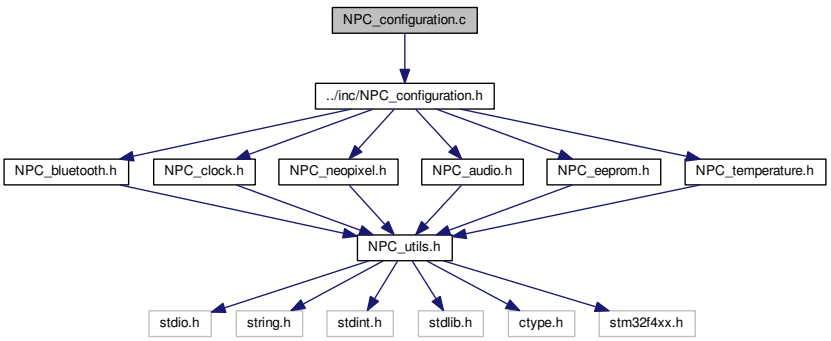
© COPYRIGHT

4.13 NPC_configuration.c File Reference

This file contains all the main initialization functions used by the NPC.

```
#include "../inc/NPC_configuration.h"
```

Include dependency graph for NPC_configuration.c:



Functions

- void NPC_init (void)

  Initialize all firmwares used by the NPC.
- void Error_Handler (void)

  This function is executed in case of error occurrence.

4.13.1 Detailed Description

This file contains all the main initialization functions used by the NPC.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

17-February-2017

Attention

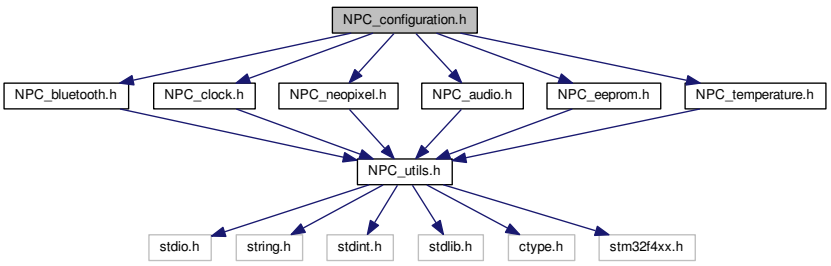
© COPYRIGHT

4.14 NPC_configuration.h File Reference

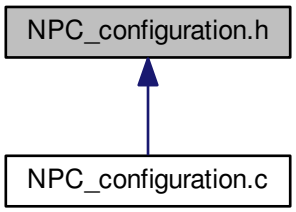
This file contains all the main initialization prototypes used by the NPC.

```
#include "NPC_bluetooth.h"
#include "NPC_clock.h"
#include "NPC_neopixel.h"
#include "NPC_audio.h"
#include "NPC_eeprom.h"
#include "NPC_temperature.h"
```

Include dependency graph for NPC_configuration.h:



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



Functions

- void NPC_init (void)

  Initialize all firmwares used by the NPC.
- void Error_Handler (void)

  This function is executed in case of error occurrence.

4.14.1 Detailed Description

This file contains all the main initialization prototypes used by the NPC.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

17-February-2017

Attention

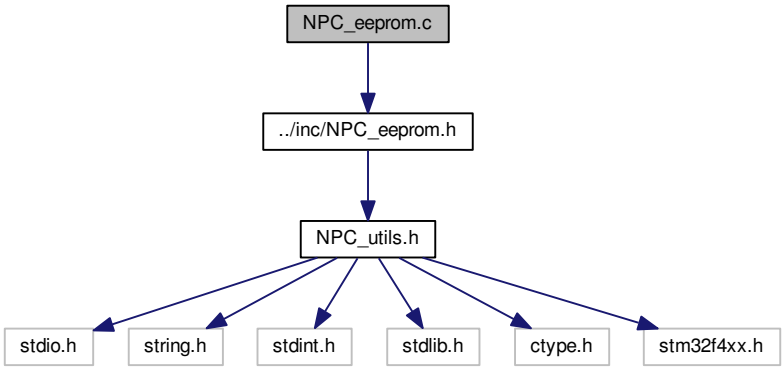
© COPYRIGHT

4.15 NPC_eeprom.c File Reference

This file provides firmware functions to manage data transmission to the eeprom.

```
#include "../inc/NPC_eeprom.h"
```

Include dependency graph for NPC_eeprom.c:



Functions

- void eeprom_init (void)

  Initialise communication to the eeprom.
- ErrorStatus eeprom_write (uint16_t address, uint8_t data)

  Write a byte to the eeprom.
- uint8_t eeprom_read (uint16_t address)

  Read a byte from the eeprom.
- ErrorStatus eeprom_write32Bytes (uint16_t baseAddress, uint8_t ∗data)

  Write a page to the eeprom.
- void eeprom_clear (void)

  Clear eeprom data.
- ErrorStatus eeprom_writeNBytes (uint16_t baseAddress, uint8_t ∗data, uint16_t N)

  Write N bytes to the eeprom.
- ErrorStatus eeprom_write4Bytes (uint16_t baseAddress, uint8_t ∗data)

  Write 4 bytes to the eeprom.
- uint32_t eeprom_read4Bytes (uint16_t baseAddress)

  Read a 32byte from eeprom.
- void eeprom_readNBytes (uint16_t baseAddress, uint8_t ∗data, uint16_t N)

4.15.1 Detailed Description

This file provides firmware functions to manage data transmission to the eeprom.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

12-March-2017

Attention

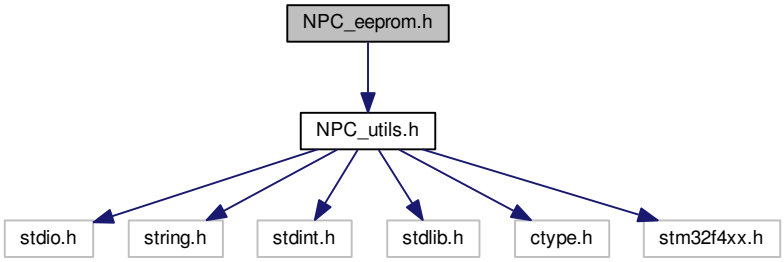
© COPYRIGHT

4.16 NPC_eeprom.h File Reference

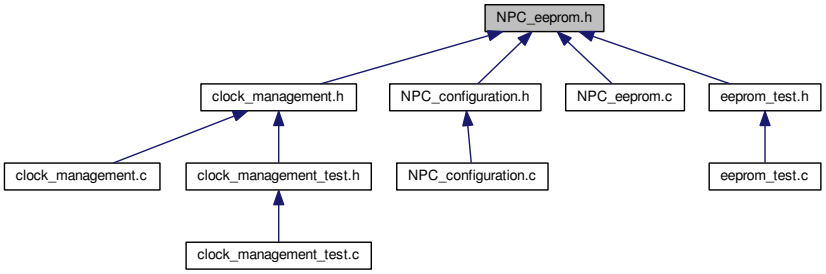
This file contains all the configuration prototypes used by the eeprom firmware.

```
#include "NPC_utils.h"
```

Include dependency graph for NPC_eeprom.h:



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



Macros

- #define WREN 0b00000110
- #define WRDI 0b00000100
- #define RDSR 0b00000101
- #define WRSR 0b00000001
- #define READ 0b00000011
- #define WRITE 0b00000010
- #define PAGE_LENGTH 32
- #define EEPROM_SIZE 0xFA00

Functions

- void eeprom_init (void)

  Initialise communication to the eeprom.
- ErrorStatus eeprom_write (uint16_t address, uint8_t data)

  Write a byte to the eeprom.
- uint8_t eeprom_read (uint16_t address)

  Read a byte from the eeprom.
- ErrorStatus eeprom_write32Bytes (uint16_t baseAddress, uint8_t ∗data)

  Write a page to the eeprom.
- uint32_t eeprom_read4Bytes (uint16_t baseAddress)

  Read a 32byte from eeprom.
- ErrorStatus eeprom_writeNBytes (uint16_t baseAddress, uint8_t ∗data, uint16_t N)

  Write N bytes to the eeprom.
- ErrorStatus eeprom_write4Bytes (uint16_t baseAddress, uint8_t ∗data)

  Write 4 bytes to the eeprom.
- void eeprom_readNBytes (uint16_t baseAddress, uint8_t ∗data, uint16_t N)
- void eeprom_clear (void)

  Clear eeprom data.

4.16.1 Detailed Description

This file contains all the configuration prototypes used by the eeprom firmware.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

12-March-2017

Attention

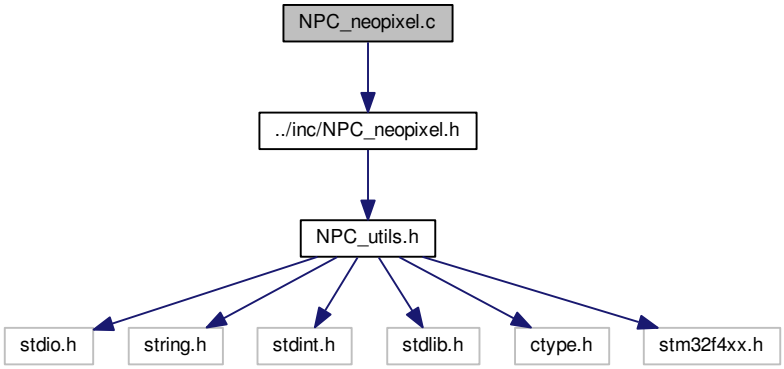
© COPYRIGHT

4.17 NPC_neopixel.c File Reference

This file provides firmware functions to manage the neopixels.

```
#include "../inc/NPC_neopixel.h"
```

Include dependency graph for NPC_neopixel.c:



Functions

- void neopixel_init (void)

  Initialise the neopixel.
- void TIM2_IRQHandler (void)

  Timer Handler for neopixel.
- void neopixel_clear (void)

  Stop pushing data to the neopixels.
- void neopixel_dataInit (void)

  Initialise the LEDbuffer.
- void neopixel_setPixelColourRGB (uint8_t n, uint8_t r, uint8_t g, uint8_t b)

  Set the colour of one led.
- void neopixel_setBrightness (uint8_t b)

  Set the brightness of the led.
- uint32_t neopixel_colourRGB (uint8_t r, uint8_t g, uint8_t b)

  convert RGB 3 8bit colour to a 32bit colour
- uint32_t neopixel_colourRGBW (uint8_t r, uint8_t g, uint8_t b, uint8_t w)

  convert RGB 3 8bit colour to a 32bit colour
- void neopixel_setPixelColourRGBW (uint8_t n, uint8_t r, uint8_t g, uint8_t b, uint8_t w)

  Set the colour of one led.
- void neopixel_setPixelColour (uint8_t n, uint32_t c)

  Set the colour of one led.

- void neopixel_setPixelColourW (uint8_t n, uint32_t c)

  Set the colour of one led.
- void neopixel_setAllPixelRGB (uint8_t r, uint8_t g, uint8_t b)

  set all the pixel on the line to a specific colour
- void neopixel_setAllPixelRGBW (uint8_t r, uint8_t g, uint8_t b, uint8_t w)

  set all the pixel on the line to a specific colour

4.17.1 Detailed Description

This file provides firmware functions to manage the neopixels.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

01-March-2017

Attention

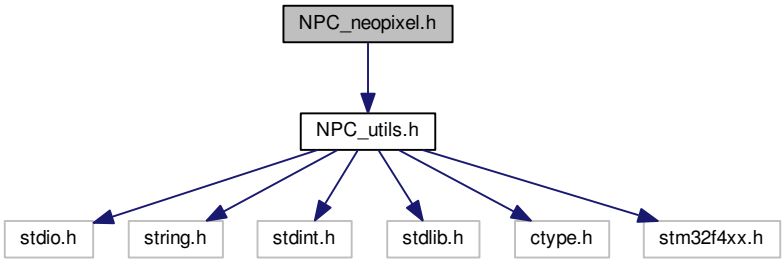
© COPYRIGHT

4.18 NPC_neopixel.h File Reference

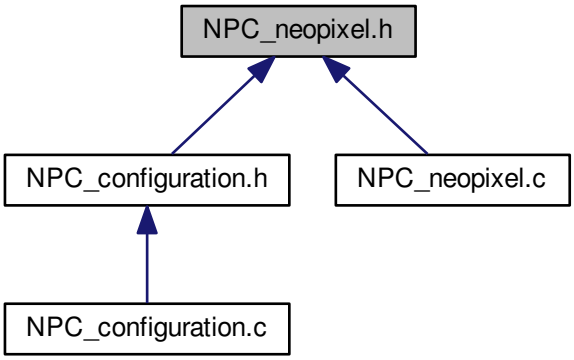
This file contains all the configuration prototypes used by the neopixel firmware.

```
#include "NPC_utils.h"
```

Include dependency graph for NPC_neopixel.h:



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



Macros

- #define WS2812_FREQ (8E5)
- #define TIMER_CLOCK_FREQ (84E6)
- #define TIMER_PERIOD (TIMER_CLOCK_FREQ / WS2812_FREQ)
- #define LED_NUMBER (4)
- #define LED_DATA_SIZE (LED_NUMBER ∗ 24)
- #define RESET_SLOTS_BEGIN (50)
- #define RESET_SLOTS_END (50)
- #define WS2812_LAST_SLOT (1)
- #define LED_BUFFER_SIZE (RESET_SLOTS_BEGIN + LED_DATA_SIZE + WS2812_LAST_SLOT + R↩ ESET_SLOTS_END)
- #define WS2812_0 (TIMER_PERIOD / 3)
- #define WS2812_1 (TIMER_PERIOD ∗ 2 / 3)
- #define WS2812_RESET (0)
- #define MAX_8BIT (255)

Functions

- void neopixel_init (void)

  Initialise the neopixel.
- void neopixel_setBrightness (uint8_t b)

  Set the brightness of the led.
- void neopixel_setState (uint8_t s)
- void neopixel_show (void)
- void neopixel_clear (void)

  Stop pushing data to the neopixels.
- void neopixel_dataInit (void)

  Initialise the LEDbuffer.
- void TIM2_IRQHandler (void)

  Timer Handler for neopixel.

- uint32_t neopixel_colourRGB (uint8_t r, uint8_t g, uint8_t b)
  convert RGB 3 8bit colour to a 32bit colour
- uint32_t neopixel_colourRGBW (uint8_t r, uint8_t g, uint8_t b, uint8_t w)
  convert RGB 3 8bit colour to a 32bit colour
- void neopixel_setPixelColourRGB (uint8_t n, uint8_t r, uint8_t g, uint8_t b)
  Set the colour of one led.
- void neopixel_setPixelColourRGBW (uint8_t n, uint8_t r, uint8_t g, uint8_t b, uint8_t w)
  Set the colour of one led.
- void neopixel_setPixelColour (uint8_t n, uint32_t c)
  Set the colour of one led.
- void neopixel_setPixelColourW (uint8_t n, uint32_t c)
  Set the colour of one led.
- void neopixel_setAllPixelRGB (uint8_t r, uint8_t g, uint8_t b)
  set all the pixel on the line to a specific colour
- void neopixel_setAllPixelRGBW (uint8_t r, uint8_t g, uint8_t b, uint8_t w)
  set all the pixel on the line to a specific colour

4.18.1 Detailed Description

This file contains all the configuration prototypes used by the neopixel firmware.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

01-March-2017

Attention

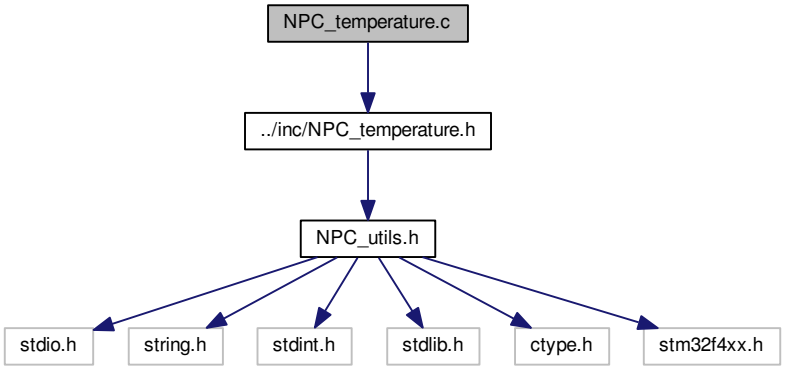
© COPYRIGHT

4.19 NPC_temperature.c File Reference

This file provides firmware functions to manage the temperature sensor.

```
#include "../inc/NPC_temperature.h"
```

Include dependency graph for NPC_temperature.c:



Functions

- void temperature_init (void)

  Initialise the ADC.
- uint16_t temperature_value (void)

  Read ADC value.
- int32_t temperature_read (void)

  Convert the ADC value to its corresponding temperature value.

4.19.1 Detailed Description

This file provides firmware functions to manage the temperature sensor.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

15-March-2017

Attention

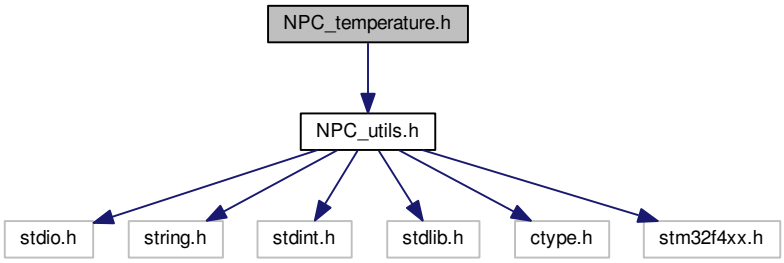
© COPYRIGHT

4.20 NPC_temperature.h File Reference

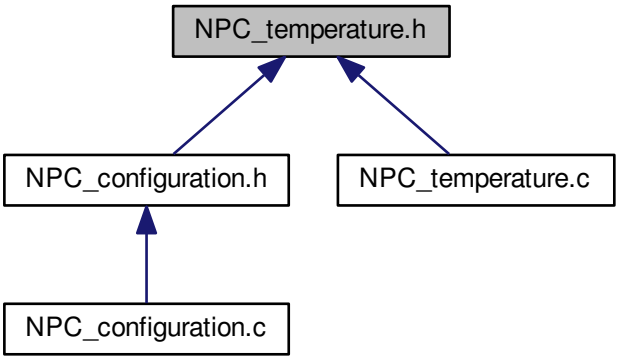
This file contains all the configuration prototypes used by the temperature firmware.

```
#include "NPC_utils.h"
```

Include dependency graph for NPC_temperature.h:



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



Functions

- void temperature_init (void)

  Initialise the ADC.
- uint16_t temperature_value (void)

  Read ADC value.
- int32_t temperature_read (void)

  Convert the ADC value to its corresponding temperature value.

4.20.1 Detailed Description

This file contains all the configuration prototypes used by the temperature firmware.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

15-March-2017

Attention

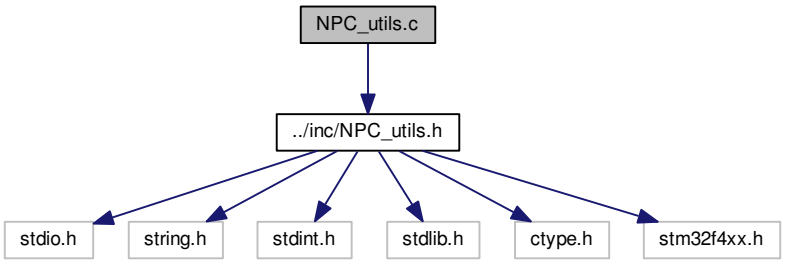
© COPYRIGHT

4.21 NPC_utils.c File Reference

This file provides utility functions to the NPC clock.

```
#include "../inc/NPC_utils.h"
```

Include dependency graph for NPC_utils.c:



Functions

- uint32_t max (uint32_t a, uint32_t b, uint32_t c)

  Find max between a, b, and c.
- void delay (uint32_t microseconds)

  delay for the number of microsecond

Variables

- uint8_t pixel_color
- uint8_t DMA_RX_Buffer [DMA_RX_BUFFER_SIZE]
- uint8_t UART_Buffer [UART_BUFFER_SIZE] = {0}

4.21.1 Detailed Description

This file provides utility functions to the NPC clock.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

23-February-2017

Attention

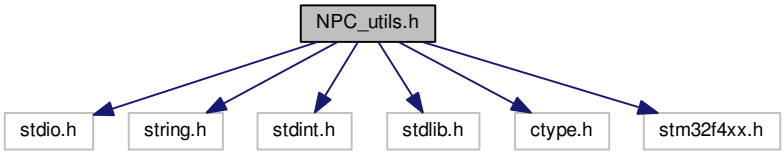
© COPYRIGHT

4.22 NPC_utils.h File Reference

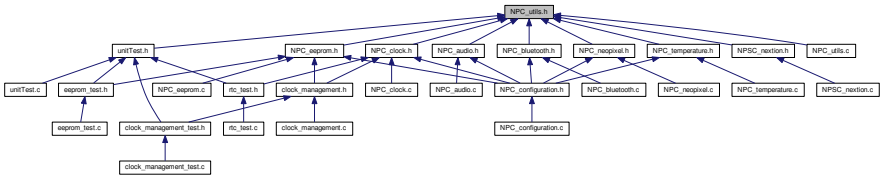
This file contains all the utility functions prototypes used by the NPC.

```
#include <stdio.h>
#include <string.h>
#include <stdint.h>
#include <stdlib.h>
#include <ctype.h>
#include "stm32f4xx.h"
```

Include dependency graph for NPC_utils.h:



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



Macros

- #define UART_BUFFER_SIZE 4
- #define DMA_RX_BUFFER_SIZE 4

Enumerations

- enum bool { false = 0, true = !false }

Functions

- uint32_t max (uint32_t a, uint32_t b, uint32_t c)

  Find max between a, b, and c.
- void delay (uint32_t microseconds)

  delay for the number of microsecond

Variables

- uint8_t pixel_color
- uint8_t DMA_RX_Buffer [DMA_RX_BUFFER_SIZE]
- uint8_t UART_Buffer [UART_BUFFER_SIZE]

4.22.1 Detailed Description

This file contains all the utility functions prototypes used by the NPC.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

23-February-2017

Attention

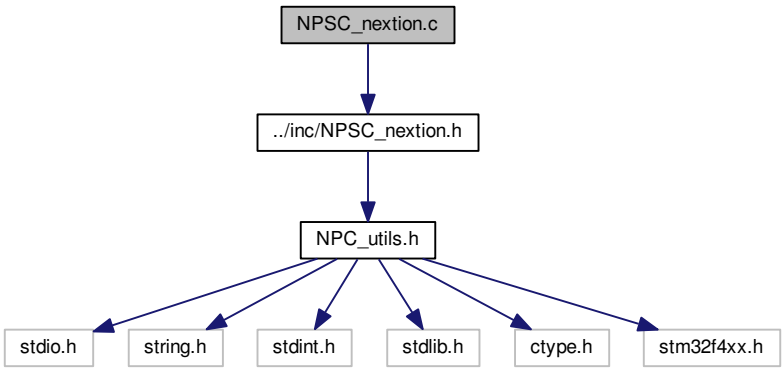
© COPYRIGHT

4.23 NPSC_nextion.c File Reference

This file provides firmware functions to manage the nextion.

```
#include "../inc/NPSC_nextion.h"
```

Include dependency graph for NPSC_nextion.c:



Functions

- void nextion_init (void)

  Initialize the nextion and set baudrate to 9600.
- void nextion_buffer_update (void)
- void USART2_IRQHandler (void)

  Global interrupt handler for USART2.
- void DMA1_Stream5_IRQHandler (void)

  Global interrupt handler for DMA1 stream5.
- void nextion_send (uint8_t *data)

  send string to the hc-06

4.23.1 Detailed Description

This file provides firmware functions to manage the nextion.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

12-October-2017

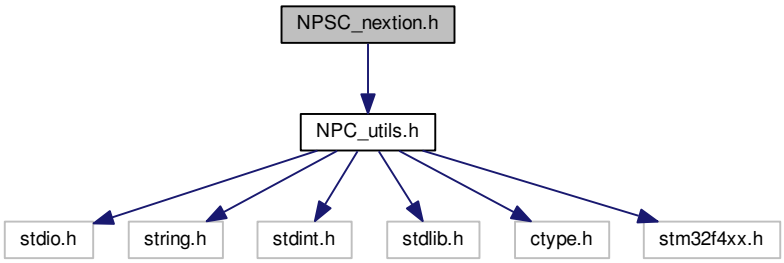
Attention

© COPYRIGHT

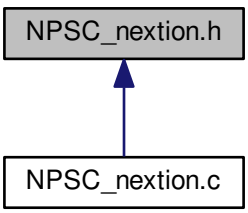
4.24 NPSC_nextion.h File Reference

```
#include "NPC_utils.h"
```

Include dependency graph for NPSC_nextion.h:



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



Macros

- #define NEXTION_PERIPH_USARTX RCC_APB1Periph_USART2
- #define NEXTION_PERIPH_GPIOX RCC_AHB1Periph_GPIOA
- #define NEXTION_GPIOX GPIOA
- #define NEXTION_TX_PIN GPIO_Pin_2
- #define NEXTION_RX_PIN GPIO_Pin_3
- #define NEXTION_TX_PINSOURCE GPIO_PinSource2
- #define NEXTION_RX_PINSOURCE GPIO_PinSource3
- #define NEXTION_AF_USART GPIO_AF_USART2
- #define NEXTION_USARTX USART2
- #define NEXTION_USARTX_IRQ USART2_IRQn
- #define NEXTION_BAUDRATE 9600

Functions

- void nextion_init (void)

  Initialize the nextion and set baudrate to 9600.
- void USART2_IRQHandler (void)

  Global interrupt handler for USART2.
- void DMA1_Stream5_IRQHandler (void)

  Global interrupt handler for DMA1 stream5.
- void nextion_send (uint8_t ∗data)

  send string to the hc-06

Variables

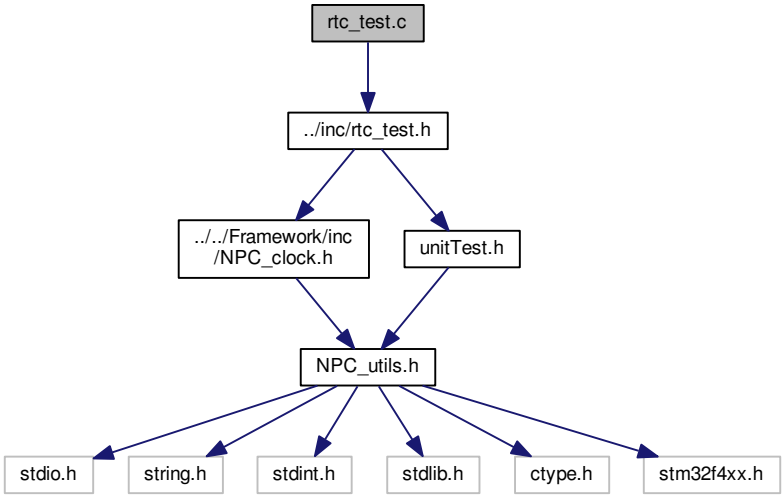
- size_t nextion_write
- size_t nextion_read

4.25 rtc_test.c File Reference

This file contains the unit test implementation for the rtc.

```
#include "../inc/rtc_test.h"
```

Include dependency graph for rtc_test.c:



Functions

- bool test_clock_date (void)

  Unit test for date save and load.
- bool test_clock_time (void)

  Unit test for time save and load.
- bool test_clock_alarm (void)

  Unit test for alarm save and load.

4.25.1 Detailed Description

This file contains the unit test implementation for the rtc.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

24-March-2017

Attention

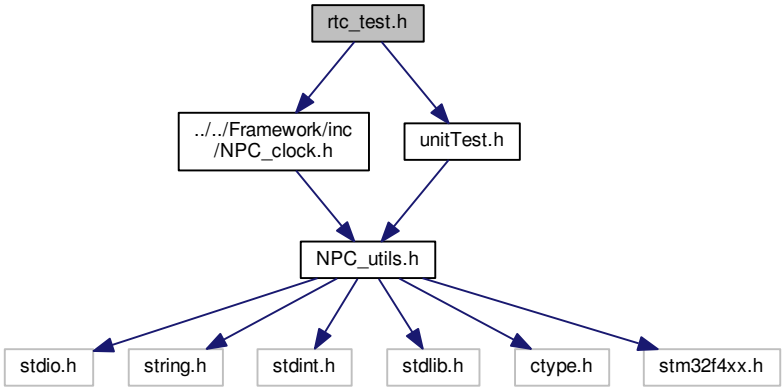
© COPYRIGHT

4.26 rtc_test.h File Reference

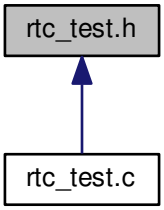
This file contains template of unit tests for the rtc.

```
#include "../../Framework/inc/NPC_clock.h"
#include "unitTest.h"
```

Include dependency graph for rtc_test.h:



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



Functions

- bool test_clock_date (void)

  Unit test for date save and load.
- bool test_clock_time (void)

  Unit test for time save and load.
- bool test_clock_alarm (void)

  Unit test for alarm save and load.

4.26.1 Detailed Description

This file contains template of unit tests for the rtc.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

24-March-2017

Attention

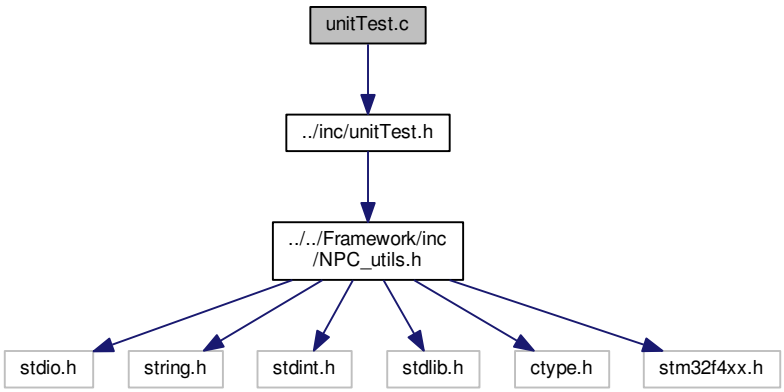
© COPYRIGHT

4.27 unitTest.c File Reference

This file contains the implementation of the function used for Unit Testing.

```
#include "../inc/unitTest.h"
```

Include dependency graph for unitTest.c:



Functions

- bool assertTrue (bool condition)

  Assert than condition is true.
- bool assertFalse (bool condition)

  Assert than condition is false.
- bool assertEqual (int a, int b)

  Assert than a equals b.
- bool assertGreater (int a, int b)

  Assert than a is greater than b.
- bool assertLess (int a, int b)

  Assert than a less than than b.
- bool assertGreaterOrEqual (int a, int b)

  Assert than a is greater or equals to b.
- bool assertLessOrEqual (int a, int b)

  Assert than a is less or equals to b.

4.27.1 Detailed Description

This file contains the implementation of the function used for Unit Testing.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

21-March-2017

Attention

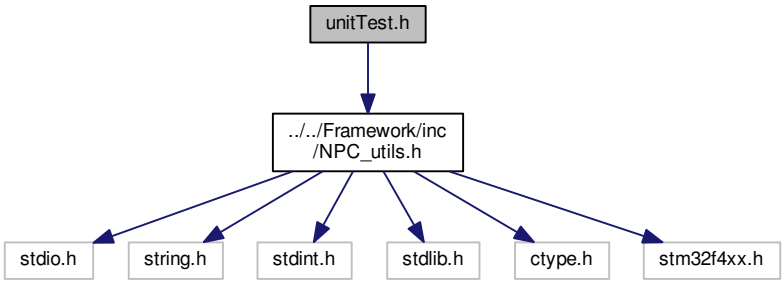
© COPYRIGHT

4.28 unitTest.h File Reference

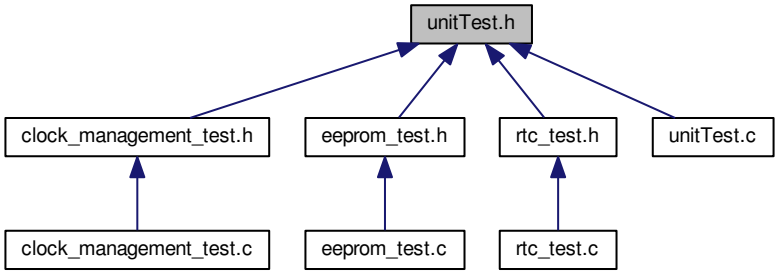
This file contains all the configuration prototypes used by the unit testing.

```
#include "../../Framework/inc/NPC_utils.h"
```

Include dependency graph for unitTest.h:



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



Functions

- bool assertTrue (bool condition)
  Assert than condition is true.
- bool assertFalse (bool condition)
  Assert than condition is false.
- bool assertEqual (int a, int b)
  Assert than a equals b.
- bool assertGreater (int a, int b)
  Assert than a is greater than b.
- bool assertLess (int a, int b)
  Assert than a less than than b.
- bool assertGreaterOrEqual (int a, int b)
  Assert than a is greater or equals to b.
- bool assertLessOrEqual (int a, int b)
  Assert than a is less or equals to b.

4.28.1 Detailed Description

This file contains all the configuration prototypes used by the unit testing.

Author

Othniel Konan (Kojey)

Version

V1.1.0

Date

21-March-2017

Attention

© COPYRIGHT