IoT Embedded Systems

Alexandre Silva v1.0

Generated by Doxygen 1.8.16

1 Module Index 1

1.1 Modules . . . 1

2 File Index 3

2.1 File List . . . 3

3 Module Documentation 5

3.1 FreeRTOS-Drivers . . . 5

3.1.1 Detailed Description . . . 5

3.2 Library information . . . 6

3.2.1 Detailed Description . . . 6

3.2.2 Function Documentation . . . 6

3.2.2.1 INFO_GetVersion() . . . 6

3.3 BH1750 . . . 7

3.3.1 Detailed Description . . . 7

3.3.2 Macro Definition Documentation . . . 8

3.3.2.1 RTOS_BH1750_QUEUE_SIZE . . . 8

3.3.2.2 RTOS_BH1750_TASK_STACK_SIZE . . . 8

3.3.3 Enumeration Type Documentation . . . 8

3.3.3.1 BH1750MeasurementTimeEnum . . . 8

3.3.3.2 BH1750ModeEnum . . . 8

3.3.4 Function Documentation . . . 10

3.3.4.1 rtosBH1750_ConfigureMode() . . . 10

3.3.4.2 rtosBH1750_GetLight() . . . 10

3.3.4.3 rtosBH1750_Init() . . . 11

3.3.4.4 rtosBH1750_Ready() . . . 11

3.3.4.5 rtosBH1750_SetMeasurementTime() . . . 11

3.4 ENCODER . . . 12

3.4.1 Detailed Description . . . 12

3.4.2 Macro Definition Documentation . . . 13

3.4.2.1 RTOS_ENCODER_QUEUE_SIZE . . . 13

3.4.2.2 RTOS_ENCODER_TASK_STACK_SIZE . . . 13

3.4.3 Enumeration Type Documentation . . . 13

3.4.3.1 ButtonEnum . . . 13

3.4.4 Function Documentation . . . 13

3.4.4.1 rtosENCODER_GetButton() . . . 13

3.4.4.2 rtosENCODER_GetValue() . . . 14

3.4.4.3 rtosENCODER_Init() . . . 14

3.5 ESP8266 serial interface . . . 15

3.5.1 Detailed Description . . . 16

3.5.2 Macro Definition Documentation . . . 16

3.5.2.1 RTOS_ESPSERIAL_BUFFER_SIZE . . . 16

3.5.2.2 RTOS_ESPSERIAL_BUFFERREFRESH_TIMEOUT . . . 16

3.5.2.3 RTOS_ESPSERIAL_DEFAULT_SETUP . . . 16
3.5.2.4 RTOS_ESPSERIAL_LONG_PAUSE . . . 16
3.5.2.5 RTOS_ESPSERIAL_PRINTF_DEBUG . . . 17
3.5.2.6 RTOS_ESPSERIAL_QUEUE_SIZE . . . 17
3.5.2.7 RTOS_ESPSERIAL_REFRESH_PERIOD_MS . . . 17
3.5.2.8 RTOS_ESPSERIAL_RESTORE_ON_DEFAULT_SETUP . . . 17
3.5.2.9 RTOS_ESPSERIAL_SETUP_TASK_STACK_SIZE . . . 17
3.5.2.10 RTOS_ESPSERIAL_SETUP_TIMEOUT . . . 17
3.5.2.11 RTOS_ESPSERIAL_SHORT_PAUSE . . . 17
3.5.3 Enumeration Type Documentation . . . 17
3.5.3.1 ESPState_t . . . 17
3.5.4 Function Documentation . . . 18
3.5.4.1 rtosESPSERIAL_Clear() . . . 18
3.5.4.2 rtosESPSERIAL_FindStr() . . . 18
3.5.4.3 rtosESPSERIAL_Init() . . . 19
3.5.4.4 rtosESPSERIAL_LockBegin() . . . 19
3.5.4.5 rtosESPSERIAL_LockEnd() . . . 19
3.5.4.6 rtosESPSERIAL_ReadLine() . . . 19
3.5.4.7 rtosESPSERIAL_Recv() . . . 20
3.5.4.8 rtosESPSERIAL_RecvByte() . . . 20
3.5.4.9 rtosESPSERIAL_Refresh() . . . 21
3.5.4.10 rtosESPSERIAL_Send() . . . 21
3.5.4.11 rtosESPSERIAL_SendAT() . . . 21
3.5.4.12 rtosESPSERIAL_Setup() . . . 22
3.5.4.13 rtosESPSERIAL_WaitResp() . . . 22
3.5.4.14 rtosESPSERIAL_WaitStr() . . . 22
3.6 Text LCD . . . 24
3.6.1 Detailed Description . . . 24
3.6.2 Macro Definition Documentation . . . 24
3.6.2.1 LCDText_COLUMNS . . . 24
3.6.2.2 LCDText_LINES . . . 24
3.6.2.3 RTOS_LCD_QUEUE_SIZE . . . 25
3.6.2.4 RTOS_LCD_TASK_STACKSIZE . . . 25
3.6.3 Function Documentation . . . 25
3.6.3.1 rtosLCDText_Clear() . . . 25
3.6.3.2 rtosLCDText_CreateChar() . . . 25
3.6.3.3 rtosLCDText_CursorOff() . . . 25
3.6.3.4 rtosLCDText_CursorOn() . . . 26
3.6.3.5 rtosLCDText_Init() . . . 26
3.6.3.6 rtosLCDText_Locate() . . . 26
3.6.3.7 rtosLCDText_LockBegin() . . . 27
3.6.3.8 rtosLCDText_LockEnd() . . . 27

3.6.3.9 rtosLCDText_Off() . . . 27
3.6.3.10 rtosLCDText_On() . . . 27
3.6.3.11 rtosLCDText_Printf() . . . 27
3.6.3.12 rtosLCDText_WriteChar() . . . 28
3.6.3.13 rtosLCDText_WriteLine() . . . 28
3.6.3.14 rtosLCDText_WriteString() . . . 28
3.7 LED . . . 29
3.7.1 Detailed Description . . . 29
3.7.2 Macro Definition Documentation . . . 29
3.7.2.1 RTOS_LED_QUEUE_SIZE . . . 29
3.7.2.2 RTOS_LED_TASK_STACK_SIZE . . . 29
3.7.3 Function Documentation . . . 29
3.7.3.1 rtosLED_GetState() . . . 29
3.7.3.2 rtosLED_Init() . . . 30
3.7.3.3 rtosLED_Off() . . . 30
3.7.3.4 rtosLED_On() . . . 30
3.7.3.5 rtosLED_Toggle() . . . 30
3.8 Task locking mechanism for FreeRTOS . . . 31
3.8.1 Detailed Description . . . 31
3.8.2 Enumeration Type Documentation . . . 31
3.8.2.1 LockState_t . . . 31
3.8.3 Function Documentation . . . 32
3.8.3.1 rtosLOCK_Begin() . . . 32
3.8.3.2 rtosLOCK_Destroy() . . . 32
3.8.3.3 rtosLOCK_End() . . . 32
3.8.3.4 rtosLOCK_Init() . . . 33
3.9 Presence detecting sensor. . . . 34
3.9.1 Detailed Description . . . 34
3.9.2 Macro Definition Documentation . . . 34
3.9.2.1 RTOS_PIR_PERIOD_MS . . . 34
3.9.3 Function Documentation . . . 34
3.9.3.1 rtosPIR_GetValue() . . . 34
3.9.3.2 rtosPIR_Init() . . . 34
3.10 Real Time Clock . . . 35
3.10.1 Detailed Description . . . 35
3.10.2 Macro Definition Documentation . . . 35
3.10.2.1 RTOS_RTC_QUEUE_SIZE . . . 35
3.10.2.2 RTOS_RTC_TASK_STACK_SIZE . . . 35
3.10.3 Function Documentation . . . 35
3.10.3.1 rtosRTC_GetSeconds() . . . 35
3.10.3.2 rtosRTC_GetValue() . . . 36
3.10.3.3 rtosRTC_Init() . . . 36

3.10.3.4 rtosRTC_InitSeconds() . . . 36
3.10.3.5 rtosRTC_SetSeconds() . . . 36
3.10.3.6 rtosRTC_SetValue() . . . 38
3.11 Timeout . . . 39
3.11.1 Detailed Description . . . 39
3.11.2 Function Documentation . . . 39
3.11.2.1 rtosTIMEOUT_Expired() . . . 39
3.11.2.2 rtosTIMEOUT_Start() . . . 39

4 File Documentation **41**

4.1 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/info.h File Reference . . . 41
4.1.1 Detailed Description . . . 41
4.2 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtosbh1750.h File Reference . . . 41
4.2.1 Detailed Description . . . 42
4.3 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtosencoder.h File Reference . . . 43
4.3.1 Detailed Description . . . 43
4.4 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtosespserial.h File Reference . . . 44
4.4.1 Detailed Description . . . 45
4.5 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtoslcd.h File Reference . . . 45
4.5.1 Detailed Description . . . 46
4.6 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtosled.h File Reference . . . 46
4.6.1 Detailed Description . . . 47
4.7 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtoslock.h File Reference . . . 47
4.7.1 Detailed Description . . . 48
4.8 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtospir.h File Reference . . . 48
4.8.1 Detailed Description . . . 48
4.9 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtosrtc.h File Reference . . . 49
4.9.1 Detailed Description . . . 49
4.10 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtostimeout.h File Reference . . . 49
4.10.1 Detailed Description . . . 50

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

FreeRTOS-Drivers 5
- Library information 6
- BH1750 7
- ENCODER 12
- ESP8266 serial interface 15
- Text LCD 24
- LED 29
- Task locking mechanism for FreeRTOS 31
- Presence detecting sensor. 34
- Real Time Clock 35
- Timeout 39

Chapter 2

File Index

2.1 File List

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

C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/info.h
Contains information about the current API version . . . 41
C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtosbh1750.h
Contains the rtosBH1750 API . . . 41
C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtosencoder.h
Contains the rtosENCODER API . . . 43
C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtosespserial.h
Contains the rtosENCODER API . . . 44
C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtoslcd.h
Contains the rtosLCDText API . . . 45
C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtosled.h
Contains the rtosLED API . . . 46
C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtoslock.h
Contains the rtosLOCK API . . . 47
C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtospir.h
Contains the rtosPIR API . . . 48
C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtosrtc.h
Contains the rtosRTC API . . . 49
C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtostimeout.h
Contains the rtosTIMEOUT API . . . 49

Chapter 3

Module Documentation

3.1 FreeRTOS-Drivers

This package is set for use with FreeRTOS-Drivers.

Modules

- Library information

  This package provides the version number of FreeRTOS-Drivers library.
- BH1750

  This package provides the capabilities to interact with the ambient light sensor BH1750 in a multithreaded manner.
- ENCODER

  This package provides the interface for driving the rotary and the push button.
- ESP8266 serial interface

  This package provides the interface for the ESP8266 with serial interface in a multi-threaded manner, along with AT command related functions.
- Text LCD

  This package provides the interface for driving 4-bit HD44780-based LCDs used in an LPCXpresso development board based on LPC1769 from NXP in a multithreaded manner.
- LED

  This package provides the capabilities to interact with the built-in LED in the LPCXpresso development board based on LPC1769 from NXP in a multi-threaded manner.
- Task locking mechanism for FreeRTOS

  This package provides the ability for a calling task to lock execution of the section of code placed in between rtosL↩ OCK_Begin and rtosLOCK_End function calls from other tasks.
- Presence detecting sensor.

  This package provides the interface for driving the presence detecting sensor in a multi-threaded manner.
- Real Time Clock

  This package provides the interface for the real time clock present in the LPC1769 microcontroller from NXP in a multi-threaded manner.
- Timeout

  This package provides the API to execute non-blocking timeouts in FreeRTOS.

3.1.1 Detailed Description

This package is set for use with FreeRTOS-Drivers.

3.2 Library information

This package provides the version number of FreeRTOS-Drivers library.

Functions

- int INFO_GetVersion (void)

3.2.1 Detailed Description

This package provides the version number of FreeRTOS-Drivers library.

3.2.2 Function Documentation

3.2.2.1 INFO_GetVersion()

```
int INFO_GetVersion (
            void  )
```

Get version number of the library.

Returns

Version number.

3.3 BH1750

This package provides the capabilities to interact with the ambient light sensor BH1750 in a multithreaded manner.

Macros

- #define RTOS_BH1750_QUEUES_SIZE 5
- #define RTOS_BH1750_TASK_STACK_SIZE configMINIMAL_STACK_SIZE

Typedefs

- typedef enum BH1750ModeEnum BH1750_ModeType

  BH1750 modes definition.
- typedef enum BH1750MeasurementTimeEnum BH1750_MeasurementTimeType

  BH1750 measurement times definition.

Enumerations

- enum BH1750ModeEnum {
  UNCONFIGURED = UNCONFIGURED, CONTINUOUS_HIGH_RES_MODE = CONTINUOUS_H↩
  IGH_RES_MODE, CONTINUOUS_HIGH_RES_MODE_2 = CONTINUOUS_HIGH_RES_MODE_↩
  2, CONTINUOUS_LOW_RES_MODE = CONTINUOUS_LOW_RES_MODE,
  ONE_TIME_HIGH_RES_MODE = ONE_TIME_HIGH_RES_MODE, ONE_TIME_HIGH_RES_MODE_2 =
  ONE_TIME_HIGH_RES_MODE_2, ONE_TIME_LOW_RES_MODE = ONE_TIME_LOW_RES_MODE }

  BH1750 modes definition.
- enum BH1750MeasurementTimeEnum { DEFAULT_MEASUREMENT_TIME = DEFAULT_MEASUREM↩
  ENT_TIME, MIN_MEASUREMENT_TIME = MIN_MEASUREMENT_TIME, MAX_MEASUREMENT_TIME =
  MAX_MEASUREMENT_TIME }

  BH1750 measurement times definition.

Functions

- bool rtosBH1750_Init (void)

  Initializes the rtosBH1750 API.
- bool rtosBH1750_ConfigureMode (BH1750_ModeType mode)

  Configures the operation mode.
- bool rtosBH1750_SetMeasurementTime (BH1750_MeasurementTimeType time)

  Configures the measurement time.
- bool rtosBH1750_Ready (bool maxWait)

  Verifies if it is possible to take measurements.
- float rtosBH1750_GetLight ()

  Reads the ambient light.

3.3.1 Detailed Description

This package provides the capabilities to interact with the ambient light sensor BH1750 in a multithreaded manner.

The sensor is connected to the microcontroller LPC1769 as shown in the follow table:

BH1750	LPC1769
SCL	P0.28
SDA	P0.27
ADDR	GND

3.3.2 Macro Definition Documentation

3.3.2.1 RTOS_BH1750_QUEUES_SIZE

```
#define RTOS_BH1750_QUEUES_SIZE 5
```

Queue size for the rtosBH1750 queues.

3.3.2.2 RTOS_BH1750_TASK_STACK_SIZE

```
#define RTOS_BH1750_TASK_STACK_SIZE configMINIMAL_STACK_SIZE
```

Minimum stack size for the BH1750 managing task.

3.3.3 Enumeration Type Documentation

3.3.3.1 BH1750MeasurementTimeEnum

```
enum BH1750MeasurementTimeEnum
```

BH1750 measurement times definition.

Enumerator

DEFAULT_MEASUREMENT_TIME	Default measurement time.
MIN_MEASUREMENT_TIME	Minimum measurement time.
MAX_MEASUREMENT_TIME	Maximum measurement time.

3.3.3.2 BH1750ModeEnum

```
enum BH1750ModeEnum
```

BH1750 modes definition.

Enumerator

UNCONFIGURED	Same as Power Down Mode
CONTINUOUS_HIGH_RES_MODE	Measurement at 1 lux resolution. Measurement time is approx 120ms.
CONTINUOUS_HIGH_RES_MODE_2	Measurement at 0.5 lux resolution. Measurement time is approx 120ms.
CONTINUOUS_LOW_RES_MODE	Measurement at 4 lux resolution. Measurement time is approx 16ms.
ONE_TIME_HIGH_RES_MODE	Measurement at 1 lux resolution. Measurement time is approx 120ms.
ONE_TIME_HIGH_RES_MODE_2	Measurement at 0.5 lux resolution. Measurement time is approx 120ms.
ONE_TIME_LOW_RES_MODE	Measurement at 4 lux resolution. Measurement time is approx 16ms.

3.3.4 Function Documentation

3.3.4.1 rtosBH1750_ConfigureMode()

```
bool rtosBH1750_ConfigureMode (
            BH1750_ModeType mode )
```

Configures the operation mode.

Parameters

mode	indicate the mode.

Returns

true if successful, false otherwise.

3.3.4.2 rtosBH1750_GetLight()

```
float rtosBH1750_GetLight ( )
```

Reads the ambient light.

Returns

Ambient light in lux.

3.3.4.3 rtosBH1750_Init()

```
bool rtosBH1750_Init (
             void  )
```

Initializes the rtosBH1750 API.

Returns

True if successful, false otherwise.

Note

This function must be called prior to any other rtosBH1750 functions.

3.3.4.4 rtosBH1750_Ready()

```
bool rtosBH1750_Ready (
             bool maxWait )
```

Verifies if it is possible to take measurements.

Parameters

maxWait	indicates the measurement time.

Returns

True if it is possible take measurements, false otherwise.

3.3.4.5 rtosBH1750_SetMeasurementTime()

```
bool rtosBH1750_SetMeasurementTime (
             BH1750_MeasurementTimeType time )
```

Configures the measurement time.

Parameters

time	indicates the measurement time.

Returns

true if successful, false otherwise.

3.4 ENCODER

This package provides the interface for driving the rotary and the push button.

Macros

- #define RTOS_ENCODER_QUEUES_SIZE 5
- #define RTOS_ENCODER_TASK_STACK_SIZE configMINIMAL_STACK_SIZE

Typedefs

- typedef enum ButtonEnum ENCODER_ButtonValueType

  Push button state structures definition.

Enumerations

- enum ButtonEnum {
  BUTTON_NOTPRESSED, BUTTON_PRESSED, BUTTON_HELD, BUTTON_RELEASE,
  BUTTON_CLICKED, BUTTON_DCLICKED }

  Push button state structures definition.

Functions

- bool rtosENCODER_Init (void)

  Initializes Encoder.
- ENCODER_ButtonValueType rtosENCODER_GetButton (void)

  Gets the value of the push button.
- ENCODER_RotationValueType rtosENCODER_GetValue (void)

  Gets the value of the rotary button.

3.4.1 Detailed Description

This package provides the interface for driving the rotary and the push button.

The rotary data bits are connected to the LPC1769 microcontroller as shown in the following table:

Rotary Button	LPC1769
CLK	P0.3
DT	P0.2
SW	P2.13
+	3V3

The driver has the following behavior: i) two steps to vary in the same direction. i) one step when an inversion is performed.

After initialization clockwise direction is assumed.

3.4.2 Macro Definition Documentation

3.4.2.1 RTOS_ENCODER_QUEUES_SIZE

```
#define RTOS_ENCODER_QUEUES_SIZE 5
```

Queue size for the rtosENCODER queues.

3.4.2.2 RTOS_ENCODER_TASK_STACK_SIZE

```
#define RTOS_ENCODER_TASK_STACK_SIZE configMINIMAL_STACK_SIZE
```

Minimum stack size for the ENCODER managing task.

3.4.3 Enumeration Type Documentation

3.4.3.1 ButtonEnum

```
enum ButtonEnum
```

Push button state structures definition.

Enumerator

BUTTON_NOTPRESSED	Button not pressed
BUTTON_PRESSED	Button pressed
BUTTON_HELD	Button held (still pressed)
BUTTON_RELEASE	Button released
BUTTON_CLICKED	Button short pressed and released
BUTTON_DCLICKED	Button double short pressed and released

3.4.4 Function Documentation

3.4.4.1 rtosENCODER_GetButton()

```
ENCODER_ButtonValueType rtosENCODER_GetButton (
            void  )
```

Gets the value of the push button.

Returns

A valid state of the push button (see ENCODER_ButtonValueType

3.4.4.2 rtosENCODER_GetValue()

```
ENCODER_RotationValueType rtosENCODER_GetValue (
            void  )
```

Gets the value of the rotary button.

Returns

0 no rotation has beed performed; returns 1 or -1 in case of clockwise and counter-clockwise rotation, respectively.

3.4.4.3 rtosENCODER_Init()

```
bool rtosENCODER_Init (
            void  )
```

Initializes Encoder.

Returns

True if successful, false otherwise

3.5 ESP8266 serial interface

This package provides the interface for the ESP8266 with serial interface in a multi-threaded manner, along with AT command related functions.

Macros

- #define RTOS_ESPSERIAL_PRINTF_DEBUG false
- #define RTOS_ESPSERIAL_DEFAULT_SETUP true
- #define RTOS_ESPSERIAL_RESTORE_ON_DEFAULT_SETUP false
- #define RTOS_ESPSERIAL_SETUP_TASK_STACK_SIZE configMINIMAL_STACK_SIZE * 3
- #define RTOS_ESPSERIAL_QUEUES_SIZE 10
- #define RTOS_ESPSERIAL_BUFFER_SIZE 512
- #define RTOS_ESPSERIAL_REFRESH_PERIOD_MS 66
- #define RTOS_ESPSERIAL_BUFFERREFRESH_TIMEOUT 500
- #define RTOS_ESPSERIAL_SHORT_PAUSE 2000
- #define RTOS_ESPSERIAL_LONG_PAUSE 10000
- #define RTOS_ESPSERIAL_SETUP_TIMEOUT 15000

Enumerations

- enum ESPState_t {
  START, QUEUE, WAIT, FAIL,
  SUCCESS }

  ESP8266 current state structure definition.

Functions

- bool rtosESPSERIAL_Init (int baudrate)

  Initializes the rtosESPSERIAL API.
- void rtosESPSERIAL_LockBegin ()

  Locks the ESP8266 device without any further action.
- void rtosESPSERIAL_LockEnd ()

  Unlocks the ESP8266 device without any further action.
- void rtosESPSERIAL_Refresh (LockState_t lock)

  Refreshes the buffer containing read data with any new incoming data.
- void rtosESPSERIAL_Clear (LockState_t lock)

  Clears the buffer containing read data.
- void rtosESPSERIAL_Send (LockState_t lock, void *command, int size, int *bytes_sent)

  Writes data to the ESP8266 device.
- void rtosESPSERIAL_Recv (LockState_t lock, uint8_t *dst_buffer, int max_bytes, int *bytes_read)

  Reads data from the ESP8266 device.
- void rtosESPSERIAL_RecvByte (LockState_t lock, uint8_t *dst_buffer, int *bytes_read)

  Reads a byte of data from the ESP8266 device.
- void rtosESPSERIAL_FindStr (LockState_t lock, char *str, char **found_str)

  Finds a desired string in the buffer containing read data.
- void rtosESPSERIAL_ReadLine (LockState_t lock, char **found_line)

  Reads a CR-LF terminated string from the ESP8266 device.

- void rtosESPSERIAL_WaitStr (LockState_t lock, char *expected_str, char **str_obtained, int timeout_ms, ESPState_t *espstate)

  Waits for a desired string to be transmitted by the ESP8266 device.
- void rtosESPSERIAL_WaitResp (LockState_t lock, char *expected_resp, char **resp_obtained, int timeout_ms, ESPState_t *espstate)

  Waits for a desired response to be transmitted by the ESP8266 device.
- void rtosESPSERIAL_SendAT (LockState_t lock, char *command, char *expected_resp, int timeout_ms, ESPState_t *espstate)

  Sends an AT command to the ESP8266 device.
- bool rtosESPSERIAL_Setup (bool restore)

  Sets up the ESP8266 device with RST/RESTORE, ATE=0, AT+CWMODE=1, AT+CWLAPOPT=1,127 and CIPMUX=0 commands.

3.5.1 Detailed Description

This package provides the interface for the ESP8266 with serial interface in a multi-threaded manner, along with AT command related functions.

3.5.2 Macro Definition Documentation

3.5.2.1 RTOS_ESPSERIAL_BUFFER_SIZE

```
#define RTOS_ESPSERIAL_BUFFER_SIZE 512
```

Buffer size incoming data transmitted by the ESP8266 device.

3.5.2.2 RTOS_ESPSERIAL_BUFFERREFRESH_TIMEOUT

```
#define RTOS_ESPSERIAL_BUFFERREFRESH_TIMEOUT 500
```

Period for the refresh function to wait for incoming data from the ESP8266 device.

3.5.2.3 RTOS_ESPSERIAL_DEFAULT_SETUP

```
#define RTOS_ESPSERIAL_DEFAULT_SETUP true
```

Calls a task to execute the setup function

3.5.2.4 RTOS_ESPSERIAL_LONG_PAUSE

```
#define RTOS_ESPSERIAL_LONG_PAUSE 10000
```

Maximum timeout in milliseconds for a long pause.

3.5.2.5 RTOS_ESPSERIAL_PRINTF_DEBUG

```
#define RTOS_ESPSERIAL_PRINTF_DEBUG false
```

Allows sent and received commands to be printed to console when set true

3.5.2.6 RTOS_ESPSERIAL_QUEUES_SIZE

```
#define RTOS_ESPSERIAL_QUEUES_SIZE 10
```

Queue size for the ESP8266 queues.

3.5.2.7 RTOS_ESPSERIAL_REFRESH_PERIOD_MS

```
#define RTOS_ESPSERIAL_REFRESH_PERIOD_MS 66
```

Execution period for rtosESPSERIAL waiting functions.

3.5.2.8 RTOS_ESPSERIAL_RESTORE_ON_DEFAULT_SETUP

```
#define RTOS_ESPSERIAL_RESTORE_ON_DEFAULT_SETUP false
```

Restores the ESP8266 restore command in the setup function

3.5.2.9 RTOS_ESPSERIAL_SETUP_TASK_STACK_SIZE

```
#define RTOS_ESPSERIAL_SETUP_TASK_STACK_SIZE configMINIMAL_STACK_SIZE * 3
```

Minimum stack size for the ESP8266 managing task.

3.5.2.10 RTOS_ESPSERIAL_SETUP_TIMEOUT

```
#define RTOS_ESPSERIAL_SETUP_TIMEOUT 15000
```

Maximum timeout in milliseconds for the setup function.

3.5.2.11 RTOS_ESPSERIAL_SHORT_PAUSE

```
#define RTOS_ESPSERIAL_SHORT_PAUSE 2000
```

Maximum timeout in milliseconds for a short pause.

3.5.3 Enumeration Type Documentation

3.5.3.1 ESPState_t

```
enum ESPState_t
```

ESP8266 current state structure definition.

Enumerator

START	Starts waiting for a desired outcome.
QUEUE	Execution state reached the implemented queue. Only used internally
WAIT	Waiting for a desired outcome.
FAIL	Desired outcome has not been achieved.
SUCCESS	Desired outcome has been achieved.

3.5.4 Function Documentation

3.5.4.1 rtosESPSERIAL_Clear()

```
void rtosESPSERIAL_Clear (
            LockState_t lock )
```

Clears the buffer containing read data.

Parameters

lock	Provides the ability to lock access to the ESP8266 device from other tasks(see LockState_t)

Note

This function must be called prior to any other rtosESPSERIAL data-receiving functions.

3.5.4.2 rtosESPSERIAL_FindStr()

```
void rtosESPSERIAL_FindStr (
            LockState_t lock,
            char * str,
            char ** found_str )
```

Finds a desired string in the buffer containing read data.

Parameters

lock	Provides the ability to lock access to the ESP8266 device from other tasks(see LockState_t)
str	Desired string to be found.
found_str	Address of the pointer where the desired string will be saved to; assigned NULL when the desired line is not found.

3.5.4.3 rtosESPSERIAL_Init()

```
bool rtosESPSERIAL_Init (
            int baudrate )
```

Initializes the rtosESPSERIAL API.

Parameters

baudrate	Set baudrate of the device.

Returns

True if successful, false otherwise.

Note

This function must be called prior to any other rtosESPSERIAL functions.

3.5.4.4 rtosESPSERIAL_LockBegin()

```
void rtosESPSERIAL_LockBegin ( )
```

Locks the ESP8266 device without any further action.

Returns

None.

3.5.4.5 rtosESPSERIAL_LockEnd()

```
void rtosESPSERIAL_LockEnd ( )
```

Unlocks the ESP8266 device without any further action.

Returns

None.

3.5.4.6 rtosESPSERIAL_ReadLine()

```
void rtosESPSERIAL_ReadLine (
            LockState_t lock,
            char ** found_line )
```

Reads a CR-LF terminated string from the ESP8266 device.

Parameters

lock	Provides the ability to lock access to the ESP8266 device from other tasks(see LockState_t)
found_line	Address of the pointer where the read line saved to; assigned NULL when the desired line is not found.

Returns

None.

3.5.4.7 rtosESPSERIAL_Recv()

```
void rtosESPSERIAL_Recv (
            LockState_t lock,
            uint8_t * dst_buffer,
            int max_bytes,
            int * bytes_read )
```

Reads data from the ESP8266 device.

Parameters

lock	Provides the ability to lock access to the ESP8266 device from other tasks(see LockState_t)
dst_buffer	Destination buffer for all data to be read into.
max_bytes	Maximum expected length for the destination buffer.
bytes_read	Address where the number of total bytes read will be saved to.

Returns

None.

3.5.4.8 rtosESPSERIAL_RecvByte()

```
void rtosESPSERIAL_RecvByte (
            LockState_t lock,
            uint8_t * dst_buffer,
            int * bytes_read )
```

Reads a byte of data from the ESP8266 device.

Parameters

lock	Provides the ability to lock access to the ESP8266 device from other tasks(see LockState_t)
dst_buffer	Destination for the read byte.
bytes_read	Address where the number of total bytes read will be saved to.

Returns

None.

3.5.4.9 rtosESPSERIAL_Refresh()

```
void rtosESPSERIAL_Refresh (
            LockState_t lock )
```

Refreshes the buffer containing read data with any new incoming data.

Parameters

lock	Provides the ability to lock access to the ESP8266 device from other tasks(see LockState_t)

Note

This function must be called prior to any other rtosESPSERIAL data-receiving functions.

3.5.4.10 rtosESPSERIAL_Send()

```
void rtosESPSERIAL_Send (
            LockState_t lock,
            void * command,
            int size,
            int * bytes_sent )
```

Writes data to the ESP8266 device.

Parameters

lock	Provides the ability to lock access to the ESP8266 device from other tasks(see LockState_t)
command	Buffer data.
size	Length of the buffer.
bytes_sent	Address where the number of total bytes written will be saved to.

Returns

None.

3.5.4.11 rtosESPSERIAL_SendAT()

```
void rtosESPSERIAL_SendAT (
            LockState_t lock,
            char * command,
            char * expected_resp,
            int timeout_ms,
            ESPState_t * espstate )
```

Sends an AT command to the ESP8266 device.

Parameters

lock	Provides the ability to lock access to the ESP8266 device from other tasks(see LockState_t)
command	Command to be sent; CR-LF is already appended to the given command by the function.
expected_resp	Expected response to be sent by the ESP8266 device.
timeout_ms	Maximum execution time for the function, in milliseconds.
espstate	Address where the current state of the ESP8266 device will be saved to. (see ESPState_t)

Returns

None.

3.5.4.12 rtosESPSERIAL_Setup()

```
bool rtosESPSERIAL_Setup (
            bool restore )
```

Sets up the ESP8266 device with RST/RESTORE, ATE=0, AT+CWMODE=1, AT+CWLAPOPT=1,127 and CIP↩MUX=0 commands.

Parameters

restore	RESTORE command will be executed

Returns

true if successful, false otherwise.

3.5.4.13 rtosESPSERIAL_WaitResp()

```
void rtosESPSERIAL_WaitResp (
            LockState_t lock,
            char * expected_resp,
            char ** resp_obtained,
            int timeout_ms,
            ESPState_t * espstate )
```

Waits for a desired response to be transmitted by the ESP8266 device.

Parameters

lock	Provides the ability to lock access to the ESP8266 device from other tasks(see LockState_t)
expected_resp	Expected response to be transmitted by the ESP8266 device.
resp_obtained	Address to the pointer to which the read line will be assigned.
timeout_ms	Maximum execution time for the function, in milliseconds.
espstate	Address where the current state of the ESP8266 device will be saved to. (see ESPState_t)

Returns

None.

3.5.4.14 rtosESPSERIAL_WaitStr()

```
void rtosESPSERIAL_WaitStr (
            LockState_t lock,
            char * expected_str,
            char ** str_obtained,
            int timeout_ms,
            ESPState_t * espstate )
```

Waits for a desired string to be transmitted by the ESP8266 device.

Parameters

lock	Provides the ability to lock access to the ESP8266 device from other tasks(see LockState_t)

Parameters

expected_str	Expected string to be transmitted by the ESP8266 device.
str_obtained	Address to the pointer to which the read string will be assigned.
timeout_ms	Maximum execution time for the function, in milliseconds.
espstate	Address where the current state of the ESP8266 device will be saved to. (see ESPState_t)

Returns

None.

3.6 Text LCD

This package provides the interface for driving 4-bit HD44780-based LCDs used in an LPCXpresso development board based on LPC1769 from NXP in a multithreaded manner.

Macros

- #define LCDText_LINES LCDText_LINES
- #define LCDText_COLUMNS LCDText_LINES
- #define RTOS_LCD_TASK_STACKSIZE configMINIMAL_STACK_SIZE * 2
- #define RTOS_LCD_QUEUE_SIZE 10

Functions

- bool rtosLCDText_Init ()
- void rtosLCDText_WriteChar (LockState_t lock, char c)
- void rtosLCDText_WriteString (LockState_t lock, const char *str)
- void rtosLCDText_WriteLine (LockState_t lock, const char *firstLine, const char *secondLine)
- void rtosLCDText_Clear (LockState_t lock)
- void rtosLCDText_Locate (LockState_t lock, int line, int column)
- void rtosLCDText_CursorOn (LockState_t lock)
- void rtosLCDText_CursorOff (LockState_t lock)
- void rtosLCDText_CreateChar (LockState_t lock, unsigned char location, unsigned char charmap[ ], int size)
- void rtosLCDText_On (LockState_t lock)
- void rtosLCDText_Off (LockState_t lock)
- void rtosLCDText_Printf (LockState_t lock, const char *fmt,...)
- void rtosLCDText_LockBegin ()
- void rtosLCDText_LockEnd ()

3.6.1 Detailed Description

This package provides the interface for driving 4-bit HD44780-based LCDs used in an LPCXpresso development board based on LPC1769 from NXP in a multithreaded manner.
The LCD data bits are connected to the microcontroller LPC1769 as shown in the following table:

LCD	LPC1769
D0 .. D3	Not connected
D4 .. D7	P2.0 .. P2.3
EN	P0.10
RS	P0.11
WR	GND

3.6.2 Macro Definition Documentation

3.6.2.1 LCDText_COLUMNS

```
#define LCDText_COLUMNS LCDText_LINES
```

LCD number of columns

3.6.2.2 LCDText_LINES

```
#define LCDText_LINES LCDText_LINES
```

LCD number of lines

3.6.2.3 RTOS_LCD_QUEUE_SIZE

```
#define RTOS_LCD_QUEUE_SIZE 10
```

Queue size for the rtosLCD queue.

3.6.2.4 RTOS_LCD_TASK_STACKSIZE

```
#define RTOS_LCD_TASK_STACKSIZE configMINIMAL_STACK_SIZE * 2
```

Minimum stack size for the LCD managing task.

3.6.3 Function Documentation

3.6.3.1 rtosLCDText_Clear()

```
void rtosLCDText_Clear (
            LockState_t lock )
```

Clears the screen and locate cursor to home position (0,0)

Parameters

lock	Provides the ability to lock access to the display from other tasks(see LockState_t)

Returns

None.

3.6.3.2 rtosLCDText_CreateChar()

```
void rtosLCDText_CreateChar (
            LockState_t lock,
            unsigned char location,
            unsigned char charmap[],
            int size )
```

User defined character.

Parameters

lock	Provides the ability to lock access to the display from other tasks(see LockState_t)
location	The new character position,
charmap	The user defined character values.
size	The size of the used defined character values buffer.

Returns

None.

3.6.3.3 rtosLCDText_CursorOff()

```
void rtosLCDText_CursorOff (
            LockState_t lock )
```

Turns cursor off.

Parameters

lock	Provides the ability to lock access to the display from other tasks(see LockState_t)

Returns

None.

3.6.3.4 rtosLCDText_CursorOn()

```
void rtosLCDText_CursorOn (
            LockState_t lock )
```

Turns the cursor on.

Parameters

lock	Provides the ability to lock access to the display from other tasks(see LockState_t)

Returns

None.

3.6.3.5 rtosLCDText_Init()

```
bool rtosLCDText_Init ( )
```

Initializes the rtosLCD API.

Returns

True if successful, false otherwise.

Note

This function must be called prior to any other rtosLCDText functions.

3.6.3.6 rtosLCDText_Locate()

```
void rtosLCDText_Locate (
            LockState_t lock,
            int line,
            int column )
```

Locates the cursor to a screen line and column

Parameters

lock	Provides the ability to lock access to the display from other tasks(see LockState_t)
line	The vertical position from the top, indexed from 0
column	The horizontal position from the left, indexed from 0

Returns

None.

3.6.3.7 rtosLCDText_LockBegin()

```
void rtosLCDText_LockBegin ( )
```

Locks the display without any other action. Particularly useful in situations where the last rtosESPSERIAL function call can vary depending on external factors (i.e. state-machines)

Returns

None.

3.6.3.8 rtosLCDText_LockEnd()

```
void rtosLCDText_LockEnd ( )
```

Unlocks the display without any other action. Particularly useful in situations where the last rtosESPSERIAL function call can vary depending on external factors (i.e. state-machines)

Returns

None.

3.6.3.9 rtosLCDText_Off()

```
void rtosLCDText_Off (
            LockState_t lock )
```

Turns the display off.

Parameters

lock	Provides the ability to lock access to the display from other tasks(see LockState_t)

Returns

None.

3.6.3.10 rtosLCDText_On()

```
void rtosLCDText_On (
            LockState_t lock )
```

Turns the display on.

Parameters

lock	Provides the ability to lock access to the display from other tasks(see LockState_t)

Returns

None.

3.6.3.11 rtosLCDText_Printf()

```
void rtosLCDText_Printf (
            LockState_t lock,
            const char * fmt,
             ... )
```

Writes a formated string to the LCD

Parameters

lock	Provides the ability to lock access to the display from other tasks(see LockState_t)
fmt	A printf-style format string, followed by the variables to use in formating the string.

3.6.3.12 rtosLCDText_WriteChar()

```
void rtosLCDText_WriteChar (
            LockState_t lock,
            char c )
```

Writes a character to the LCD

Parameters

lock	Provides the ability to lock access to the display from other tasks(see LockState_t)
c	The character to write to the display

3.6.3.13 rtosLCDText_WriteLine()

```
void rtosLCDText_WriteLine (
            LockState_t lock,
            const char * firstLine,
            const char * secondLine )
```

Writes a C-string to specified line of the LCD

Parameters

lock	Provides the ability to lock access to the display from other tasks(see LockState_t)
firstLine	The C-string to write to the first of display. If NULL nothing is write.
secondLine	The C-string to write to the second of display. If NULL nothing is write.

Returns

None.

3.6.3.14 rtosLCDText_WriteString()

```
void rtosLCDText_WriteString (
            LockState_t lock,
            const char * str )
```

Writes a C-string to the LCD

Parameters

lock	Provides the ability to lock access to the display from other tasks(see LockState_t)
str	The C-string to write to the display

3.7 LED

This package provides the capabilities to interact with the built-in LED in the LPCXpresso development board based on LPC1769 from NXP in a multi-threaded manner.

Macros

- #define RTOS_LED_QUEUES_SIZE 5
- #define RTOS_LED_TASK_STACK_SIZE configMINIMAL_STACK_SIZE

Functions

- bool rtosLED_Init (bool state)

  Initializes the rtosLED API.
- bool rtosLED_GetState (void)

  Get the current LED state.
- void rtosLED_On (void)

  Turn the LED on.
- void rtosLED_Off (void)

  Turn the LED off.
- void rtosLED_Toggle (void)

  Toggle the LED.

3.7.1 Detailed Description

This package provides the capabilities to interact with the built-in LED in the LPCXpresso development board based on LPC1769 from NXP in a multi-threaded manner.

3.7.2 Macro Definition Documentation

3.7.2.1 RTOS_LED_QUEUES_SIZE

```
#define RTOS_LED_QUEUES_SIZE 5
```

Queue size for the rtosLED queues.

3.7.2.2 RTOS_LED_TASK_STACK_SIZE

```
#define RTOS_LED_TASK_STACK_SIZE configMINIMAL_STACK_SIZE
```

Minimum stack size for the LED managing task.

3.7.3 Function Documentation

3.7.3.1 rtosLED_GetState()

```
bool rtosLED_GetState (
            void  )
```

Get the current LED state.

Returns

status of LED: "false" indicate LED is off and "true" LED is on.

3.7.3.2 rtosLED_Init()

```
bool rtosLED_Init (
            bool state )
```

Initializes the rtosLED API.

Parameters

state	set LED state: "false" turns LED off and "true" turns LED on.

Returns

True if successful, false otherwise.

Note

This function must be called prior to any other rtosLED functions. The LED will be started with the value passed in the state parameter.

3.7.3.3 rtosLED_Off()

```
void rtosLED_Off (
            void  )
```

Turn the LED off.

Returns

None.

3.7.3.4 rtosLED_On()

```
void rtosLED_On (
            void  )
```

Turn the LED on.

Returns

None.

3.7.3.5 rtosLED_Toggle()

```
void rtosLED_Toggle (
            void  )
```

Toggle the LED.

Returns

None.

3.8 Task locking mechanism for FreeRTOS

This package provides the ability for a calling task to lock execution of the section of code placed in between rtosLOCK_Begin and rtosLOCK_End function calls from other tasks.

Typedefs

- typedef struct LockStruct * LockHandle_t

  Lock handle structure definition.

Enumerations

- enum LockState_t {
  LOCK_NONE, LOCK_BEGIN, LOCK_CONTINUE, LOCK_END,
  LOCK_SINGLE }

  Lock state definition. LOCK_NONE or LOCK_SINGLE can be executed independently. LOCK_BEGIN and LOCK_↩ END should be called in the beginning and end, respectively, of a code section to be locked, with LOCK_CONTINUE being used in between the last two, if needed.

Functions

- LockHandle_t rtosLOCK_Init ()

  Initializes the rtosLOCK API.
- void rtosLOCK_Begin (LockState_t lock, LockHandle_t handle)

  Allows the beginning of a code section lock for an initialized lock handle.
- void rtosLOCK_End (LockState_t lock, LockHandle_t handle)

  Allows the end of a code section lock for an initialized lock handle.
- void rtosLOCK_Destroy (LockHandle_t handle)

  Destroys an initialized lock handle.

3.8.1 Detailed Description

This package provides the ability for a calling task to lock execution of the section of code placed in between rtosLOCK_Begin and rtosLOCK_End function calls from other tasks.

3.8.2 Enumeration Type Documentation

3.8.2.1 LockState_t

`enum LockState_t`

Lock state definition. LOCK_NONE or LOCK_SINGLE can be executed independently. LOCK_BEGIN and LOC↩ K_END should be called in the beginning and end, respectively, of a code section to be locked, with LOCK_CON↩ TINUE being used in between the last two, if needed.

Enumerator

LOCK_NONE	Ignores the locking mechanism
LOCK_BEGIN	Begins locking of a code section
LOCK_CONTINUE	Continues locking of a code section
LOCK_END	Ends locking of a code section
LOCK_SINGLE	Begins and ends locking of the code section in a single call

3.8.3 Function Documentation

3.8.3.1 rtosLOCK_Begin()

```
void rtosLOCK_Begin (
            LockState_t lock,
            LockHandle_t handle )
```

Allows the beginning of a code section lock for an initialized lock handle.

Parameters

lock	Provides the ability to lock access to the code section until rtosLOCK_End function call(see LockState_t)
handle	Lock handle (see LockHandle_t)

Returns

None.

Note

Undefined behaviour whenever the lock sequence is not executed in the proper order.

3.8.3.2 rtosLOCK_Destroy()

```
void rtosLOCK_Destroy (
            LockHandle_t handle )
```

Destroys an initialized lock handle.

Parameters

handle	Lock handle (see LockHandle_t)

Returns

None.

3.8.3.3 rtosLOCK_End()

```
void rtosLOCK_End (
            LockState_t lock,
            LockHandle_t handle )
```

Allows the end of a code section lock for an initialized lock handle.

Parameters

lock	Provides the ability to end the locking of the code section previously begun by the rtosLOCK_Begin function call(see LockState_t)
handle	Lock handle (see LockHandle_t)

Returns

None.

Note

Undefined behaviour whenever the lock sequence is not executed in the proper order.

3.8.3.4 rtosLOCK_Init()

```
LockHandle_t rtosLOCK_Init ( )
```

Initializes the rtosLOCK API.

Returns

Lock handle upon success, NULL otherwise.

3.9 Presence detecting sensor.

This package provides the interface for driving the presence detecting sensor in a multi-threaded manner.

Macros

- #define RTOS_PIR_PERIOD_MS 100

Functions

- bool rtosPIR_Init ()

  Initializes the PIR Presence detecting sensor.
- bool rtosPIR_GetValue ()

  Gets the most recent presence detection value.

3.9.1 Detailed Description

This package provides the interface for driving the presence detecting sensor in a multi-threaded manner.
The sensor is connected to the microcontroller LPC1769 in the P2.12 pin.

3.9.2 Macro Definition Documentation

3.9.2.1 RTOS_PIR_PERIOD_MS

```
#define RTOS_PIR_PERIOD_MS 100
```

Period in milliseconds to update the presence detection value.

3.9.3 Function Documentation

3.9.3.1 rtosPIR_GetValue()

```
bool rtosPIR_GetValue ( )
```

Gets the most recent presence detection value.

Returns

True if presence was detected, false otherwise.

3.9.3.2 rtosPIR_Init()

```
bool rtosPIR_Init ( )
```

Initializes the PIR Presence detecting sensor.

Returns

True if successful, false otherwise.

3.10 Real Time Clock

This package provides the interface for the real time clock present in the LPC1769 microcontroller from NXP in a multi-threaded manner.

Macros

- #define RTOS_RTC_QUEUES_SIZE 5
- #define RTOS_RTC_TASK_STACK_SIZE configMINIMAL_STACK_SIZE

Functions

- bool rtosRTC_Init (struct tm ∗dateTime)

  Initializes rtosRTC API and starts counting.
- bool rtosRTC_InitSeconds (time_t time)

  Initializes rtosRTC and starts counting.
- void rtosRTC_GetValue (struct tm ∗dateTime)

  Gets date and time from RTC.
- void rtosRTC_SetValue (struct tm ∗dateTime)

  Sets date and time to RTC.
- time_t rtosRTC_GetSeconds (void)

  Gets date and time from RTC.
- void rtosRTC_SetSeconds (time_t time)

  Sets date and time from RTC.

3.10.1 Detailed Description

This package provides the interface for the real time clock present in the LPC1769 microcontroller from NXP in a multi-threaded manner.

3.10.2 Macro Definition Documentation

3.10.2.1 RTOS_RTC_QUEUES_SIZE

```
#define RTOS_RTC_QUEUES_SIZE 5
```

Queue size for the rtosRTC queues.

3.10.2.2 RTOS_RTC_TASK_STACK_SIZE

```
#define RTOS_RTC_TASK_STACK_SIZE configMINIMAL_STACK_SIZE
```

Minimum stack size for the RTC managing task.

3.10.3 Function Documentation

3.10.3.1 rtosRTC_GetSeconds()

```
time_t rtosRTC_GetSeconds (
            void  )
```

Gets date and time from RTC.

Returns

A C standard time_t with the number of seconds since 01.01.1970 00:00:00

3.10.3.2 rtosRTC_GetValue()

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

Gets date and time from RTC.

Parameters

**dateTime*	A pointer to C standard structure tm to save data to.

Returns

None.

3.10.3.3 rtosRTC_Init()

```
bool rtosRTC_Init (
            struct tm * dateTime )
```

Initializes rtosRTC API and starts counting.

Parameters

dateTime	A pointer to C standard structure tm.

Note

If you power off the LPCXpresso board the RTC will stop.

Returns

None.

3.10.3.4 rtosRTC_InitSeconds()

```
bool rtosRTC_InitSeconds (
            time_t time )
```

Initializes rtosRTC and starts counting.

Parameters

time	A C standard time_t value.

Note

If you use rtosRTC_Init, do not use this function.

Returns

None.

3.10.3.5 rtosRTC_SetSeconds()

```
void rtosRTC_SetSeconds (
            time_t time )
```

Sets date and time from RTC.

Parameters

time	A C standard time_t with the number of seconds since 01.01.1970 00:00:00

3.10.3.6 rtosRTC_SetValue()

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

Sets date and time to RTC.

Parameters

∗dateTime	A pointer to C standard structure tm with date and time

Returns

None.

3.11 Timeout

This package provides the API to execute non-blocking timeouts in FreeRTOS.

Functions

- TickType_t rtosTIMEOUT_Start ()

  Initializes a timeout.
- bool rtosTIMEOUT_Expired (TickType_t start_time, TickType_t timeout)

  Checks if a timeout has occurred.

3.11.1 Detailed Description

This package provides the API to execute non-blocking timeouts in FreeRTOS.

3.11.2 Function Documentation

3.11.2.1 rtosTIMEOUT_Expired()

```
bool rtosTIMEOUT_Expired (
            TickType_t start_time,
            TickType_t timeout )
```

Checks if a timeout has occurred.

Parameters

start_time	Tick ammount of a previously initiated timeout.
timeout	Timeout period to check for, in ticks.

Returns

True whenever a timeout has occurred, false otherwise.

3.11.2.2 rtosTIMEOUT_Start()

```
TickType_t rtosTIMEOUT_Start ( )
```

Initializes a timeout.

Returns

Starting tick ammount.

Chapter 4

File Documentation

4.1 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/info.h File Reference

Contains information about the current API version.

Functions

- int INFO_GetVersion (void)

4.1.1 Detailed Description

Contains information about the current API version.

Version

1.0

Date

17 jun 2023

Author

Alexandre Silva

Copyright(C) 2023, Alexandre Silva All rights reserved.
Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the products. This software is supplied "AS IS" without any warranties.

4.2 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-↩ Drivers/inc/rtosbh1750.h File Reference

Contains the rtosBH1750 API.

```
#include "bh1750.h"
#include <stdint.h>
#include <stdbool.h>
```

Macros

- #define RTOS_BH1750_QUEUES_SIZE 5
- #define RTOS_BH1750_TASK_STACK_SIZE configMINIMAL_STACK_SIZE

Typedefs

- typedef enum BH1750ModeEnum BH1750_ModeType

  BH1750 modes definition.

- typedef enum BH1750MeasurementTimeEnum BH1750_MeasurementTimeType

  BH1750 measurement times definition.

Enumerations

- enum BH1750ModeEnum {
  UNCONFIGURED = UNCONFIGURED, CONTINUOUS_HIGH_RES_MODE = CONTINUOUS_H↩
  IGH_RES_MODE, CONTINUOUS_HIGH_RES_MODE_2 = CONTINUOUS_HIGH_RES_MODE_↩
  2, CONTINUOUS_LOW_RES_MODE = CONTINUOUS_LOW_RES_MODE,
  ONE_TIME_HIGH_RES_MODE = ONE_TIME_HIGH_RES_MODE, ONE_TIME_HIGH_RES_MODE_2 =
  ONE_TIME_HIGH_RES_MODE_2, ONE_TIME_LOW_RES_MODE = ONE_TIME_LOW_RES_MODE }

  BH1750 modes definition.

- enum BH1750MeasurementTimeEnum { DEFAULT_MEASUREMENT_TIME = DEFAULT_MEASUREM↩
  ENT_TIME, MIN_MEASUREMENT_TIME = MIN_MEASUREMENT_TIME, MAX_MEASUREMENT_TIME =
  MAX_MEASUREMENT_TIME }

  BH1750 measurement times definition.

Functions

- bool rtosBH1750_Init (void)

  Initializes the rtosBH1750 API.

- bool rtosBH1750_ConfigureMode (BH1750_ModeType mode)

  Configures the operation mode.

- bool rtosBH1750_SetMeasurementTime (BH1750_MeasurementTimeType time)

  Configures the measurement time.

- bool rtosBH1750_Ready (bool maxWait)

  Verifies if it is possible to take measurements.

- float rtosBH1750_GetLight ()

  Reads the ambient light.

4.2.1 Detailed Description

Contains the rtosBH1750 API.

Version

1.0

Date

17 jun 2023

Author

Alexandre Silva

Copyright(C) 2023, Alexandre Silva All rights reserved.
Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the products. This software is supplied "AS IS" without any warranties.

4.3 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtosencoder.h File Reference

Contains the rtosENCODER API.

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

Macros

- #define RTOS_ENCODER_QUEUES_SIZE 5
- #define RTOS_ENCODER_TASK_STACK_SIZE configMINIMAL_STACK_SIZE

Typedefs

- typedef enum ButtonEnum ENCODER_ButtonValueType

  Push button state structures definition.

Enumerations

- enum ButtonEnum {
  BUTTON_NOTPRESSED, BUTTON_PRESSED, BUTTON_HELD, BUTTON_RELEASE,
  BUTTON_CLICKED, BUTTON_DCLICKED }

  Push button state structures definition.

Functions

- bool rtosENCODER_Init (void)

  Initializes Encoder.
- ENCODER_ButtonValueType rtosENCODER_GetButton (void)

  Gets the value of the push button.
- ENCODER_RotationValueType rtosENCODER_GetValue (void)

  Gets the value of the rotary button.

4.3.1 Detailed Description

Contains the rtosENCODER API.

Version

1.0

Date

17 jun 2023

Author

Alexandre Silva

Copyright(C) 2023, Alexandre Silva All rights reserved.
Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the products. This software is supplied "AS IS" without any warranties.

4.4 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-↩ Drivers/inc/rtosespserial.h File Reference

Contains the rtosENCODER API.

```
#include <stdbool.h>
#include <freertos.h>
#include <time.h>
#include "rtoslock.h"
```

Macros

- #define RTOS_ESPSERIAL_PRINTF_DEBUG false
- #define RTOS_ESPSERIAL_DEFAULT_SETUP true
- #define RTOS_ESPSERIAL_RESTORE_ON_DEFAULT_SETUP false
- #define RTOS_ESPSERIAL_SETUP_TASK_STACK_SIZE configMINIMAL_STACK_SIZE * 3
- #define RTOS_ESPSERIAL_QUEUES_SIZE 10
- #define RTOS_ESPSERIAL_BUFFER_SIZE 512
- #define RTOS_ESPSERIAL_REFRESH_PERIOD_MS 66
- #define RTOS_ESPSERIAL_BUFFERREFRESH_TIMEOUT 500
- #define RTOS_ESPSERIAL_SHORT_PAUSE 2000
- #define RTOS_ESPSERIAL_LONG_PAUSE 10000
- #define RTOS_ESPSERIAL_SETUP_TIMEOUT 15000

Enumerations

- enum ESPState_t {
  START, QUEUE, WAIT, FAIL,
  SUCCESS }

  ESP8266 current state structure definition.

Functions

- bool rtosESPSERIAL_Init (int baudrate)

  Initializes the rtosESPSERIAL API.
- void rtosESPSERIAL_LockBegin ()

  Locks the ESP8266 device without any further action.
- void rtosESPSERIAL_LockEnd ()

  Unlocks the ESP8266 device without any further action.
- void rtosESPSERIAL_Refresh (LockState_t lock)

  Refreshes the buffer containing read data with any new incoming data.
- void rtosESPSERIAL_Clear (LockState_t lock)

  Clears the buffer containing read data.
- void rtosESPSERIAL_Send (LockState_t lock, void *command, int size, int *bytes_sent)

  Writes data to the ESP8266 device.
- void rtosESPSERIAL_Recv (LockState_t lock, uint8_t *dst_buffer, int max_bytes, int *bytes_read)

  Reads data from the ESP8266 device.
- void rtosESPSERIAL_RecvByte (LockState_t lock, uint8_t *dst_buffer, int *bytes_read)

  Reads a byte of data from the ESP8266 device.
- void rtosESPSERIAL_FindStr (LockState_t lock, char *str, char **found_str)

  Finds a desired string in the buffer containing read data.
- void rtosESPSERIAL_ReadLine (LockState_t lock, char **found_line)

  Reads a CR-LF terminated string from the ESP8266 device.

- void rtosESPSERIAL_WaitStr (LockState_t lock, char *expected_str, char **str_obtained, int timeout_ms, ESPState_t *espstate)

  Waits for a desired string to be transmitted by the ESP8266 device.
- void rtosESPSERIAL_WaitResp (LockState_t lock, char *expected_resp, char **resp_obtained, int timeout_ms, ESPState_t *espstate)

  Waits for a desired response to be transmitted by the ESP8266 device.
- void rtosESPSERIAL_SendAT (LockState_t lock, char *command, char *expected_resp, int timeout_ms, ESPState_t *espstate)

  Sends an AT command to the ESP8266 device.
- bool rtosESPSERIAL_Setup (bool restore)

  Sets up the ESP8266 device with RST/RESTORE, ATE=0, AT+CWMODE=1, AT+CWLAPOPT=1,127 and CIPMUX=0 commands.

4.4.1 Detailed Description

Contains the rtosENCODER API.

Version

1.0

Date

17 jun 2023

Author

Alexandre Silva

Copyright(C) 2023, Alexandre Silva All rights reserved.
Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the products. This software is supplied "AS IS" without any warranties.

4.5 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtoslcd.h File Reference

Contains the rtosLCDText API.

```
#include <stdint.h>
#include <stdbool.h>
#include "rtoslock.h"
#include "lcdtext.h"
```

Macros

- #define LCDText_LINES LCDText_LINES
- #define LCDText_COLUMNS LCDText_LINES
- #define RTOS_LCD_TASK_STACKSIZE configMINIMAL_STACK_SIZE * 2
- #define RTOS_LCD_QUEUE_SIZE 10

Functions

- bool rtosLCDText_Init ()
- void rtosLCDText_WriteChar (LockState_t lock, char c)
- void rtosLCDText_WriteString (LockState_t lock, const char *str)
- void rtosLCDText_WriteLine (LockState_t lock, const char *firstLine, const char *secondLine)
- void rtosLCDText_Clear (LockState_t lock)

- void rtosLCDText_Locate (LockState_t lock, int line, int column)
- void rtosLCDText_CursorOn (LockState_t lock)
- void rtosLCDText_CursorOff (LockState_t lock)
- void rtosLCDText_CreateChar (LockState_t lock, unsigned char location, unsigned char charmap[ ], int size)
- void rtosLCDText_On (LockState_t lock)
- void rtosLCDText_Off (LockState_t lock)
- void rtosLCDText_Printf (LockState_t lock, const char ∗fmt,...)
- void rtosLCDText_LockBegin ()
- void rtosLCDText_LockEnd ()

4.5.1 Detailed Description

Contains the rtosLCDText API.

Version

1.0

Date

17 jun 2023

Author

Alexandre Silva

Copyright(C) 2023, Alexandre Silva All rights reserved.
Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the products. This software is supplied "AS IS" without any warranties.

4.6 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-↩ Drivers/inc/rtosled.h File Reference

Contains the rtosLED API.

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

Macros

- #define RTOS_LED_QUEUES_SIZE 5
- #define RTOS_LED_TASK_STACK_SIZE configMINIMAL_STACK_SIZE

Functions

- bool rtosLED_Init (bool state)

  Initializes the rtosLED API.
- bool rtosLED_GetState (void)

  Get the current LED state.
- void rtosLED_On (void)

  Turn the LED on.
- void rtosLED_Off (void)

  Turn the LED off.
- void rtosLED_Toggle (void)

  Toggle the LED.

4.6.1 Detailed Description

Contains the rtosLED API.

Version

1.0

Date

17 jun 2023

Author

Alexandre Silva

Copyright(C) 2023, Alexandre Silva All rights reserved.
Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the products. This software is supplied "AS IS" without any warranties.

4.7 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-↩ Drivers/inc/rtoslock.h File Reference

Contains the rtosLOCK API.

```
#include <stdbool.h>
#include <FreeRTOS.h>
#include <task.h>
#include <semphr.h>
```

Typedefs

- typedef struct LockStruct * LockHandle_t

  Lock handle structure definition.

Enumerations

- enum LockState_t {
  LOCK_NONE, LOCK_BEGIN, LOCK_CONTINUE, LOCK_END,
  LOCK_SINGLE }

  Lock state definition. LOCK_NONE or LOCK_SINGLE can be executed independently. LOCK_BEGIN and LOCK_↩ END should be called in the beginning and end, respectively, of a code section to be locked, with LOCK_CONTINUE being used in between the last two, if needed.

Functions

- LockHandle_t rtosLOCK_Init ()

  Initializes the rtosLOCK API.

- void rtosLOCK_Begin (LockState_t lock, LockHandle_t handle)

  Allows the beginning of a code section lock for an initialized lock handle.

- void rtosLOCK_End (LockState_t lock, LockHandle_t handle)

  Allows the end of a code section lock for an initialized lock handle.

- void rtosLOCK_Destroy (LockHandle_t handle)

  Destroys an initialized lock handle.

4.7.1 Detailed Description

Contains the rtosLOCK API.

Version

1.0

Date

17 jun 2023

Author

Alexandre Silva

Copyright(C) 2023, Alexandre Silva All rights reserved.
Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the products. This software is supplied "AS IS" without any warranties.

4.8 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-↩ Drivers/inc/rtospir.h File Reference

Contains the rtosPIR API.

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

Macros

- #define RTOS_PIR_PERIOD_MS 100

Functions

- bool rtosPIR_Init ()

  Initializes the PIR Presence detecting sensor.
- bool rtosPIR_GetValue ()

  Gets the most recent presence detection value.

4.8.1 Detailed Description

Contains the rtosPIR API.

Version

1.0

Date

17 jun 2023

Author

Alexandre Silva

Copyright(C) 2023, Alexandre Silva All rights reserved.
Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the products. This software is supplied "AS IS" without any warranties.

4.9 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtosrtc.h File Reference

Contains the rtosRTC API.

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

Macros

- #define RTOS_RTC_QUEUES_SIZE 5
- #define RTOS_RTC_TASK_STACK_SIZE configMINIMAL_STACK_SIZE

Functions

- bool rtosRTC_Init (struct tm *dateTime)

  Initializes rtosRTC API and starts counting.
- bool rtosRTC_InitSeconds (time_t time)

  Initializes rtosRTC and starts counting.
- void rtosRTC_GetValue (struct tm *dateTime)

  Gets date and time from RTC.
- void rtosRTC_SetValue (struct tm *dateTime)

  Sets date and time to RTC.
- time_t rtosRTC_GetSeconds (void)

  Gets date and time from RTC.
- void rtosRTC_SetSeconds (time_t time)

  Sets date and time from RTC.

4.9.1 Detailed Description

Contains the rtosRTC API.

Version

1.0

Date

17 jun 2023

Author

Alexandre Silva

Copyright(C) 2023, Alexandre Silva All rights reserved.
Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the products. This software is supplied "AS IS" without any warranties.

4.10 C:/Users/alext/Desktop/SE/Workspace/FreeRTOS-Drivers/inc/rtostimeout.h File Reference

Contains the rtosTIMEOUT API.

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

Functions

- TickType_t rtosTIMEOUT_Start ()

  Initializes a timeout.
- bool rtosTIMEOUT_Expired (TickType_t start_time, TickType_t timeout)

  Checks if a timeout has occurred.

4.10.1 Detailed Description

Contains the rtosTIMEOUT API.

Version

1.0

Date

12 jul 2023

Author

Alexandre Silva

Copyright(C) 2023, Alexandre Silva All rights reserved.
Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the products. This software is supplied "AS IS" without any warranties.