



**ASR IoT Series**

# **RTOS Application Notes**

**Version 1.0.0**

**Issue Date 2023-08-29**

**Copyright © 2023 ASR**

## About This Document

This document introduces the usage of all the lega\_rtos APIs of ASR IoT series Wi-Fi+BLE Combo SoC and Wi-Fi SoC chips.

## Intended Readers

This document is mainly for engineers who use this chip to develop their own platforms and products, for instance:

- PCB Hardware Development Engineer
- Software Engineer
- Technical Support Engineer

## Included Chip Models

This document applies to ASR IoT series Wi-Fi+BLE Combo SoC and Wi-Fi SoC chips.

## Copyright Notice

© 2023 ASR Microelectronics Co., Ltd. All rights reserved. No part of this document can be reproduced, transmitted, transcribed, stored, or translated into any language in any form or by any means without the written permission of ASR Microelectronics Co., Ltd.

## Trademark Statement



ASR and ASR Microelectronics Co., Ltd. are trademarks of ASR Microelectronics Co., Ltd.

Other trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners and are hereby declared.

## Electrostatic Discharge (ESD) Warning

This product can be damaged by Electrostatic Discharge (ESD). When handling with this device, the people should be very careful to conduct the ESD protection to avoid any device damage caused by ESD event.

## Disclaimer

ASR does not give any warranty of any kind and may make improvements and/or changes in this document or in the product described in this document at any time.

This document is only used as a guide, and no contents in the document constitute any form of warranty. Information in this document is subject to change without notice.

All liability, including liability for infringement of any proprietary rights caused by using the information in this document, is disclaimed.

## ASR Microelectronics Co., Ltd.

**Address:** 9F, Building 10, No. 399 Keyuan Road, Zhangjiang High-tech Park, Pudong New Area, Shanghai, 201203, China

**Homepage:** <http://www.asrmicro.com/>

## Revision History

Date	Version	Release Notes
2023.08	V1.0.0	First release.

# Table of Contents

<b>1. Introduction .....</b>	<b>1</b>
<b>2. Interrupts .....</b>	<b>2</b>
2.1 Functional Description .....	2
2.2 APIs .....	2
2.2.1 lega_rtos_enter_critical .....	2
2.2.2 lega_rtos_exit_critical .....	2
2.2.3 lega_rtos_is_in_interrupt_context .....	3
<b>3. Thread Control .....</b>	<b>4</b>
3.1 Functional Description .....	4
3.2 APIs .....	4
3.2.1 lega_rtos_create_thread .....	4
3.2.2 lega_rtos_delete_thread .....	5
3.2.3 lega_rtos_suspend_thread .....	5
3.2.4 lega_rtos_is_current_thread .....	5
3.2.5 lega_rtos_get_current_thread .....	5
3.2.6 lega_rtos_delay_millisecond .....	6
3.2.7 lega_rtos_thread_force_awake .....	6
3.2.8 lega_rtos_print_thread_status .....	6
<b>4. Semaphores .....</b>	<b>7</b>
4.1 Functional Description .....	7
4.2 APIs .....	7
4.2.1 lega_rtos_init_semaphore .....	7
4.2.2 lega_rtos_set_semaphore .....	7
4.2.3 lega_rtos_get_semaphore .....	8
4.2.4 lega_rtos_deinit_semaphore .....	8
4.2.5 lega_rtos_semaphore_pending_thread_null .....	8
<b>5. Mutex .....</b>	<b>9</b>
5.1 Functional Description .....	9
5.2 APIs .....	9
5.2.1 lega_rtos_init_mutex .....	9
5.2.2 lega_rtos_lock_mutex .....	9
5.2.3 lega_rtos_unlock_mutex .....	10
5.2.4 lega_rtos_deinit_mutex .....	10
<b>6. Queues .....</b>	<b>11</b>
6.1 Functional Description .....	11
6.2 APIs .....	11
6.2.1 lega_rtos_init_queue .....	11
6.2.2 lega_rtos_push_to_queue .....	11

6.2.3	lega_rtos_pop_from_queue .....	12
6.2.4	lega_rtos_deinit_queue .....	12
6.2.5	lega_rtos_is_queue_empty .....	13
6.2.6	lega_rtos_is_queue_full .....	13
<b>7.</b>	<b>Time .....</b>	<b>14</b>
7.1	Functional Description .....	14
7.2	APIs .....	14
7.2.1	lega_rtos_get_time .....	14
7.2.2	lega_rtos_get_system_ticks .....	14
<b>8.</b>	<b>Timers .....</b>	<b>15</b>
8.1	Functional Description .....	15
8.2	APIs .....	15
8.2.1	lega_rtos_init_timer .....	15
8.2.2	lega_rtos_start_timer .....	15
8.2.3	lega_rtos_stop_timer .....	16
8.2.4	lega_rtos_reload_timer .....	16
8.2.5	lega_rtos_deinit_timer .....	16
8.2.6	lega_rtos_is_timer_running .....	16
<b>9.</b>	<b>Event Groups .....</b>	<b>17</b>
9.1	Functional Description .....	17
9.2	APIs .....	17
9.2.1	lega_rtos_init_event_flags .....	17
9.2.2	lega_rtos_wait_for_event_flags .....	17
9.2.3	lega_rtos_set_event_flags .....	18
9.2.4	lega_rtos_deinit_event_flags .....	18
<b>10.</b>	<b>Utility .....</b>	<b>19</b>
10.1	Functional Description .....	19
10.2	APIs .....	19
10.2.1	lega_rtos_get_system_version .....	19
10.2.2	lega_rtos_task_cfg_get .....	19
<b>11.</b>	<b>Memory .....</b>	<b>20</b>
11.1	Functional Description .....	20
11.2	APIs .....	20
11.2.1	lega_rtos_malloc .....	20
11.2.2	lega_rtos_free .....	20

# 1.

# Introduction

ASR lega\_rtos is a wrapper interface for mainstream RTOS, and tested RTOS are FreeRTOS, AliOS-rhino, and HUAWEI-liteOS. We recommend user application to use this layer, considering the compatibility of ASR\_SDK on different RTOS platforms.

All function interfaces are declared in lega\_rtos\_api.h.

ASR Confidential

## 2. Interrupts

### 2.1 Functional Description

LEGA\_RTOS\_Interrupt Operations.

### 2.2 APIs

#### 2.2.1 lega\_rtos\_enter\_critical

<b>Function</b>	Enters a critical session, all interrupts are disabled.
<b>Definition/Command</b>	void lega_rtos_enter_critical(void)
<b>Parameters</b>	None
<b>Return</b>	None
<b>Notes</b>	
<b>Example</b>	<pre>int32_t ret = 0; lega_rtos_enter_critical(); ret = duet_flash_erase(PARTITION_PARAMETER_2,                       0, KV_MAX_SIZE); lega_rtos_exit_critical();</pre>

#### 2.2.2 lega\_rtos\_exit\_critical

<b>Function</b>	Exits a critical session, all interrupts are enabled.
<b>Definition/Command</b>	void lega_rtos_enter_critical(void)
<b>Parameters</b>	None
<b>Return</b>	None
<b>Notes</b>	
<b>Example</b>	<pre>int32_t ret = 0; lega_rtos_enter_critical(); ret = duet_flash_erase(PARTITION_PARAMETER_2,                       0, KV_MAX_SIZE); lega_rtos_exit_critical();</pre>

### 2.2.3 lega\_rtos\_is\_in\_interrupt\_context

<b>Function</b>	Judges whether in interrupt context or not.
<b>Definition/Command</b>	OSBool lega_rtos_is_in_interrupt_context(void)
<b>Parameters</b>	None
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>● TRUE: In an interrupt context.</li> <li>● FALSE: Not in an interrupt context.</li> </ul>
<b>Notes</b>	
<b>Example</b>	<pre> if (lega_rtos_is_in_interrupt_context() == FALSE) {     threadENTER_CRITICAL(); } else {     threadENTER_CRITICAL_FROM_ISR(); } </pre>



# 3. Thread Control

## 3.1 Functional Description

LEGA RTOS thread management functions provide thread creation, delete, suspend, resume, and other RTOS management APIs.

## 3.2 APIs

### 3.2.1 lega\_rtos\_create\_thread

<b>Function</b>	Creates an application thread that starts an execution at the specified thread entry function.
<b>Definition/Command</b>	OSStatus lega_rtos_create_thread(lega_thread_t *thread, uint8_t priority, const char *name, lega_thread_function_t function, int32_t stack_size, lega_thread_arg_t arg)
<b>Parameters</b>	<p>thread: A pointer to the user-supplied thread handler.</p> <p>priority: Specifies a priority value between 0 and 255. The lower the numeric value, the higher the thread's priority.</p> <p>name: A pointer to name for the thread.</p> <p>function: The main thread entry function.</p> <p>stack_size: Specifies the number of bytes in the stack.</p> <p>arg: An argument which will be passed to the thread function.</p>
<b>Return</b>	<p>Result:</p> <ul style="list-style-type: none"> <li>• kNoErr: Indicates the successful completion of the service.</li> <li>• kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	<pre>/* creates a thread with the thread handler mThread, with a name MyThread, with an entry function EventLoopThreadMain, with a priority 3, with a stack size 1024, with an argument NULL */  OSStatus status = lega_rtos_create_thread(&amp;mThread, 3,     "MyThread", (lega_thread_function_t)EventLoopThreadMain,     1024, NULL);</pre>

### 3.2.2 lega\_rtos\_delete\_thread

<b>Function</b>	This service deletes a terminated thread.
<b>Definition/Command</b>	OSStatus lega_rtos_delete_thread(lega_thread_t *thread)
<b>Parameters</b>	thread: A pointer to the user supplied thread handler UNLL is the current thread.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>• kNoErr: Indicates the successful completion of the service.</li> <li>• kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	/* deletes current thread */ lega_rtos_delete_thread(NULL);

### 3.2.3 lega\_rtos\_suspend\_thread

<b>Function</b>	This service suspends the specified application thread. A thread may call this service to suspend itself.
<b>Definition/Command</b>	void lega_rtos_suspend_thread(lega_thread_t *thread)
<b>Parameters</b>	thread: The handler of the thread to suspend, NULL is the current thread.
<b>Return</b>	None
<b>Notes</b>	
<b>Example</b>	/* suspends AT_task thread */ lega_rtos_suspend_thread(&AT_task);

### 3.2.4 lega\_rtos\_is\_current\_thread

<b>Function</b>	Tells whether if the input thread is the current running thread.
<b>Definition/Command</b>	OSBool lega_rtos_is_current_thread(lega_thread_t *thread)
<b>Parameters</b>	thread: A thread handler to be tested.
<b>Return</b>	Status: <ul style="list-style-type: none"> <li>• TRUE: Specified thread is the current thread.</li> <li>• FALSE: Specified thread is not currently running.</li> </ul>
<b>Notes</b>	
<b>Example</b>	

### 3.2.5 lega\_rtos\_get\_current\_thread

<b>Function</b>	Gets the current running application thread handler pointer.
<b>Definition/Command</b>	lega_thread_t * lega_rtos_get_current_thread(void)
<b>Parameters</b>	None
<b>Return</b>	<ul style="list-style-type: none"> <li>• This service returns a current running thread handler pointer.</li> </ul>
<b>Notes</b>	
<b>Example</b>	

### 3.2.6 lega\_rtos\_delay\_millisecond

<b>Function</b>	This service causes the calling thread to suspend for the specified number of milliseconds. This service can be called only from an application thread.
<b>Definition/Command</b>	OSStatus lega_rtos_delay_milliseconds(uint32_t num_ms)
<b>Parameters</b>	num_ms: Milliseconds
<b>Return</b>	<ul style="list-style-type: none"> <li>This service call always returns success.</li> </ul>
<b>Notes</b>	
<b>Example</b>	<pre> while (true) {     OSStatus status;     /* Check the event queue. */     if (lega_rtos_is_queue_empty(&amp;sAppEventQueue))     {         lega_rtos_delay_milliseconds(1);         continue;     } } </pre>

### 3.2.7 lega\_rtos\_thread\_force\_awake

<b>Function</b>	Forcibly wakes another thread. Causes the specified thread to wake from suspension. This will usually cause an error or timeout in that thread, since the task it is waiting is not completed.
<b>Definition/Command</b>	OSStatus lega_rtos_thread_force_awake(lega_thread_t *thread)
<b>Parameters</b>	thread: The handle of the other thread which will be woken.
<b>Return</b>	<b>Result:</b> <ul style="list-style-type: none"> <li>kNoErr: Indicates successful completion of the service.</li> <li>kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	

### 3.2.8 lega\_rtos\_print\_thread\_status

<b>Function</b>	Prints thread status into buffer.
<b>Definition/Command</b>	OSStatus lega_rtos_print_thread_status(char* buffer, int length)
<b>Parameters</b>	buffer: A pointer to a buffer to store the thread status. length: Length of the buffer.
<b>Return</b>	<ul style="list-style-type: none"> <li>kNoErr: This service call always returns success.</li> </ul>
<b>Notes</b>	
<b>Example</b>	

# 4. Semaphores

## 4.1 Functional Description

LEGA RTOS semaphore functions provide management APIs for semaphores such as init,set,get and dinit.

## 4.2 APIs

### 4.2.1 lega\_rtos\_init\_semaphore

<b>Function</b>	This service creates a counting semaphore for the inter-thread synchronization. The initial semaphore count is specified as an input parameter.
<b>Definition/Command</b>	OSStatus lega_rtos_init_semaphore(lega_semaphore_t * semaphore, int value)
<b>Parameters</b>	semaphore: A pointer to a semaphore handler. value: Specifies the initial count for this semaphore.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>• kNoErr: Indicates successful completion of the service.</li> <li>• KGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	lega_semaphore_t at_cmd_protect; lega_rtos_init_semaphore(&at_cmd_protect, 0);

### 4.2.2 lega\_rtos\_set\_semaphore

<b>Function</b>	Sets (post/put/increasement) a semaphore.
<b>Definition/Command</b>	OSStatus lega_rtos_set_semaphore(lega_semaphore_t * semaphore)
<b>Parameters</b>	semaphore: A pointer to the previously created a counting semaphore.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>• kNoErr: Indicates the successful completion of the service.</li> <li>• kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	lega_rtos_set_semaphore(&at_cmd_protect);

### 4.2.3 lega\_rtos\_get\_semaphore

<b>Function</b>	This service retrieves an instance (a single count) from the specified counting semaphore. As a result, the specified semaphore's count is decreased by one. If semaphore is at zero already, then the calling thread will be suspended until another thread sets the semaphore with <code>lega_rtos_set_semaphore</code> .
<b>Definition/Command</b>	<code>OSStatus lega_rtos_get_semaphore(lega_semaphore_t *semaphore, uint32_t timeout_ms)</code>
<b>Parameters</b>	<code>semaphore</code> : A pointer to a previously created counting semaphore handler. <code>timeout_ms</code> : The number of milliseconds to wait before returning.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>• <code>kNoErr</code>: Indicates the successful completion of the service.</li> <li>• <code>kGeneralErr</code>: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	<pre> if (lega_rtos_get_semaphore(&amp;at_cmd_protect,                            AT_CMD_TIMEOUT)) {     printf("pre cmd is running\n"); } </pre>

### 4.2.4 lega\_rtos\_deinit\_semaphore

<b>Function</b>	This service deletes the specified counting semaphore created with <code>lega_rtos_init_semaphore</code> .
<b>Definition/Command</b>	<code>OSStatus lega_rtos_deinit_semaphore(lega_semaphore * semaphore)</code>
<b>Parameters</b>	<code>semaphore</code> : A pointer to a previously created semaphore.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>• <code>kNoErr</code>: Indicates the successful completion of the service.</li> <li>• <code>kGeneralErr</code>: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	

### 4.2.5 lega\_rtos\_semaphore\_pending\_thread\_null

<b>Function</b>	Tells if the thread number pending by this semaphore is 0.
<b>Definition/Command</b>	<code>OSBool lega_rtos_semaphore_pending_thread_null(lega_semaphore_t * semaphore)</code>
<b>Parameters</b>	<code>semaphore</code> : A pointer to a previously created semaphore handler.
<b>Return</b>	result: <ul style="list-style-type: none"> <li>• <code>TRUE</code>: Pending thread number is 0</li> <li>• <code>FALSE</code>: Pending thread number is not 0</li> </ul>
<b>Notes</b>	
<b>Example</b>	

# 5.

# Mutex

## 5.1 Functional Description

LEGA RTOS mutex functions provide management APIs for a mutex such as init, lock, unlock and dinit.

## 5.2 APIs

### 5.2.1 lega\_rtos\_init\_mutex

<b>Function</b>	This service creates a mutex for an inter-thread mutual exclusion for the resource protection.
<b>Definition/Command</b>	OSStatus lega_rtos_init_mutex(lega_mutex_t *mutex)
<b>Parameters</b>	mutex: A pointer to a mutex handler.
<b>Return</b>	result: <ul style="list-style-type: none"> <li>• kNoErr: Indicates the successful completion of the service.</li> <li>• KGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	<pre>lega_mutex_t LwIPCoreLock; lega_rtos_init_mutex(&amp;LwIPCoreLock);</pre>

### 5.2.2 lega\_rtos\_lock\_mutex

<b>Function</b>	This service attempts to obtain the exclusive ownership of the specified mutex. If the mutex is already held by another thread, the calling thread will be suspended until the mutex lock is released by the other thread.
<b>Definition/Command</b>	OSStatus lega_rtos_lock_mutex(lega_mutex_t *mutex, uint32_t timeout_ms)
<b>Parameters</b>	mutex: A pointer to the mutex handler. timeout_ms: NO_WAIT/WAIT_FOREVER/milliseconds.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>• kNoErr: Indicates successful completion of the service.</li> <li>• kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	<pre>lega_rtos_lock_mutex(&amp;LwIPCoreLock, LEGA_WAIT_FOREVER);</pre>

### 5.2.3 lega\_rtos\_unlock\_mutex

<b>Function</b>	Releases a currently held mutex. If another thread is waiting on the mutex lock, then it will be resumed.
<b>Definition/Command</b>	OSStatus lega_rtos_unlock_mutex(lega_mutex_t *mutex)
<b>Parameters</b>	mutex: A pointer to the mutex handler.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>• kNoErr: Indicates the successful completion of the service.</li> <li>• kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	lega_rtos_unlock_mutex(&LwIPCoreLock);

### 5.2.4 lega\_rtos\_deinit\_mutex

<b>Function</b>	This service deletes the specified mutex created with lega_rtos_init_mutex.
<b>Definition/Command</b>	OSStatus lega_rtos_deinit_mutex(lega_mutex_t *mutex)
<b>Parameters</b>	mutex: A pointer to a previously created mutex handler.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>• kNoErr: Indicates the successful completion of the service.</li> <li>• kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	

# 6. Queues

## 6.1 Functional Description

LEGARTOS FIFO queue functions provide management APIs for FIFO such as init, push, pop and dinit.

## 6.2 APIs

### 6.2.1 lega\_rtos\_init\_queue

<b>Function</b>	This service creates a message queue that is typically used for the inter-thread communication.
<b>Definition/Command</b>	OSStatus lega_rtos_init_queue(lega_queue_t *queue, const char *name, uint32_t message_size, uint32_t number_of_message)
<b>Parameters</b>	queue: A pointer of the queue handler. name: A pointer to the name of the message queue. message_size: Specifies the size of each message in the queue. number_of_message: Number of the message –i.e. max number of messages in the queue.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>• kNoErr: Indicates the successful completion of the service.</li> <li>• kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	lega_queue_t at_task_msg_queue; lega_rtos_init_queue(&at_task_msg_queue, "AT_TASK_QUEUE", sizeof(uint32_t), AT_QUEUE_SIZE);

### 6.2.2 lega\_rtos\_push\_to\_queue

<b>Function</b>	This service sends a message to the specified message queue.
<b>Definition/Command</b>	OSStatus lega_rtos_push_to_queue(lega_queue_t *queue, void *message, uint32_t timeout_ms)
<b>Parameters</b>	queue: A pointer to the queue handler. message: A pointer to the message. timeout_ms: TX_NO_WAIT/TX_WAIT_FOREVER/timeout value Selecting TX_NO_WAIT results in an immediate return from this service regardless of whether or not it is successful. Selecting TX_WAIT_FOREVER causes the calling thread to suspend indefinitely until there is a room in the queue.



	Selecting a numeric value specifies the maximum number of timer-ticks to stay suspending while waiting for the room in the queue.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>● kNoErr: Indicates a successful completion of the service.</li> <li>● kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	lega_rtos_push_to_queue(&at_task_msg_queue, &at_msg, LEGA_NO_WAIT);

### 6.2.3 lega\_rtos\_pop\_from\_queue

<b>Function</b>	This service retrieves a message from the specified message queue.
<b>Definition/Command</b>	OSStatus lega_rtos_pop_from_queue(lega_queue_t *queue, void *message, uint32_t timeout_ms)
<b>Parameters</b>	queue: A pointer to the queue handler. message: A pointer to a buffer that will receive the object being popped off the queue. Size is assumed to be the size specified in lega_rtos_init_queue, hence you must ensure the buffer is long enough or memory corruption will result. timeout_ms: The number of milliseconds to wait before returning.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>● kNoErr: Indicates a successful completion of the service.</li> <li>● kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	lega_rtos_pop_from_queue(&at_task_msg_queue, &msg_queue_elmt, LEGA_WAIT_FOREVER);

### 6.2.4 lega\_rtos\_deinit\_queue

<b>Function</b>	This service deletes a queue created with lega_rtos_init_queue.
<b>Definition/Command</b>	OSStatus lega_rtos_deinit_queue(lega_queue_t *queue)
<b>Parameters</b>	queue: A pointer to the specified queue handler.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>● kNoErr: Indicates the successful completion of the service.</li> <li>● kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	lega_rtos_deinit_queue(&at_task_msg_queue);

### 6.2.5 lega\_rtos\_is\_queue\_empty

<b>Function</b>	Checks if a queue is empty.
<b>Definition/Command</b>	OSBool lega_rtos_is_queue_empty(lega_queue_t *queue)
<b>Parameters</b>	queue: A pointer to the queue handler.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>• TRUE: A queue is empty.</li> <li>• FALSE: A queue is not empty.</li> </ul>
<b>Notes</b>	
<b>Example</b>	<pre> if (lega_rtos_is_queue_empty(&amp;mEventQueue)) {     return; } </pre>

### 6.2.6 lega\_rtos\_is\_queue\_full

<b>Function</b>	Checks if a queue is full.
<b>Definition/Command</b>	OSBool lega_rtos_is_queue_full(lega_queue_t *queue)
<b>Parameters</b>	queue: A pointer to the queue handler.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>• TRUE: A queue is full.</li> <li>• FALSE: A queue is not full.</li> </ul>
<b>Notes</b>	
<b>Example</b>	

# 7. Time

## 7.1 Functional Description

LEGA RTOS time functions.

## 7.2 APIs

### 7.2.1 lega\_rtos\_get\_time

<b>Function</b>	Gets time in milliseconds since the RTOS starts. Since this is only 32 bits, it will roll over every 49 days, 17 hours.
<b>Definition/Command</b>	uint32_t lega_rtos_get_time(void)
<b>Parameters</b>	None
<b>Return</b>	This service returns time in milliseconds since RTOS starts.
<b>Notes</b>	
<b>Example</b>	

### 7.2.2 lega\_rtos\_get\_system\_ticks

<b>Function</b>	This service returns the contents of the internal system timer tick. Each timer-tick increases the internal system clock by one. The system clock is set to zero during initialization.
<b>Definition/Command</b>	uint32_t lega_rtos_get_system_ticks(void)
<b>Parameters</b>	None
<b>Return</b>	The value of the system ticks since RTOS starts.
<b>Notes</b>	
<b>Example</b>	

# 8. Timers

## 8.1 Functional Description

LEGA RTOS timer functions provide management APIs for timer such as init,start,stop,reload and dinit.

## 8.2 APIs

### 8.2.1 lega\_rtos\_init\_timer

<b>Function</b>	This service creates an application timer with the specified expiration function and period. Timer does not start running until lega_start_timer is called
<b>Definition/Command</b>	OSStatus lega_rtos_init_timer(lega_timer_t *timer, uint32_t time_ms, timer_handler_t function, void *arg)
<b>Parameters</b>	timer: A pointer to the timer handler. time_ms: A timer period in milliseconds. function: An application function to call when the timer expires. arg: An argument will be passed to the timer expired function.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>• kNoErr: Indicates the successful completion of the service.</li> <li>• kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	lega_timer_t mTimer; lega_rtos_init_timer(&mTimer, 1000, (timer_handler_t)TimerExpiredFunction, NULL);

### 8.2.2 lega\_rtos\_start\_timer

<b>Function</b>	This service activates the specified application timer.
<b>Definition/Command</b>	OSStatus lega_rtos_start_timer(lega_timer_t *timer)
<b>Parameters</b>	timer: A pointer to the timer handler to start.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>• kNoErr: Indicates the successful completion of the service.</li> <li>• kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	OSStatus status = lega_rtos_start_timer(&mTimer);

### 8.2.3 lega\_rtos\_stop\_timer

<b>Function</b>	This service deactivates the specified application timer.
<b>Definition/Command</b>	OSStatus lega_rtos_stop_timer(lega_tiemr_t *timer)
<b>Parameters</b>	timer: A pointer to the timer handler to stop.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>• kNoErr: Indicates the successful completion of the service.</li> <li>• kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	OSStatus status = lega_rtos_stop_timer(&mTimer);

### 8.2.4 lega\_rtos\_reload\_timer

<b>Function</b>	Reloads a RTOS timer that has expired. This is usually called in the timer callback handler, to reschedule the timer for the next period.
<b>Definition/Command</b>	OSStatus lega_rtos_reload_timer(lega_timer_t *timer)
<b>Parameters</b>	timer: A pointer to the timer handler to reload.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>• kNoErr: Indicates the successful completion of the service.</li> <li>• kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	

### 8.2.5 lega\_rtos\_deinit\_timer

<b>Function</b>	Deletes a RTOS timer created with lega_rtos_init_timer.
<b>Definition/Command</b>	OSStatus lega_rtos_deinit_timer(lega_timer_t *timer)
<b>Parameters</b>	timer: A pointer to the timer handler to delete.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>• kNoErr: Indicates the successful completion of the service.</li> <li>• kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	

### 8.2.6 lega\_rtos\_is\_timer\_running

<b>Function</b>	Checks if an RTOS timer is running.
<b>Definition/Command</b>	OSBool lega_rtos_is_timer_running(lega_timer_t *timer)
<b>Parameters</b>	timer: A pointer to the timer handler to check.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>• TRUE: The timer is running.</li> <li>• FALSE: The timer is not running.</li> </ul>
<b>Notes</b>	
<b>Example</b>	<pre> If (lega_rtos_is_timer_running(&amp;mTimer)==TRUE) {     // do something } </pre>

# 9. Event Groups

## 9.1 Functional Description

LEGA RTOS event group functions.

## 9.2 APIs

### 9.2.1 lega\_rtos\_init\_event\_flags

<b>Function</b>	This service creates a group of 32 event flags. All 32 event flags in the group are initialized to zero. Each event flag is represented by a single bit.
<b>Definition/Command</b>	OSStatus lega_rtos_init_event_flags(lega_event_flags_t *event_flags)
<b>Parameters</b>	event_flags: A pointer to the event group handler.
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>• kNoErr: Indicates the successful completion of the service.</li> <li>• kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	lega_event_flags_t mEventFlags; OSStatus status= lega_rtos_init_event_flags(&mEventFlags);

### 9.2.2 lega\_rtos\_wait\_for\_event\_flags

<b>Function</b>	This service retrieves event flags from the specified event flags group. Each event flags group contains 32 event flags. Each flag is represented by a single bit.
<b>Definition/Command</b>	OSStatus lega_rtos_wait_for_event_flags( <div style="text-align: right;">lega_event_flags_t            *event_flags, uint32_t</div> <div style="text-align: right;">flags_to_wait_for, uint32_t *flags_set, OSBool</div> <div style="text-align: right;">clear_set_flags, lega_event_flags_wait_option_t</div> <div style="text-align: right;">wait_option, uint32_t timeout_ms)</div>
<b>Parameters</b>	event_flags: A pointer to an event_flags handler. flags_to_wait_for: A 32-bit unsigned variable that represents the requested event flags. flags_set: Returns actual event group flags. clear_set_flags: A clear event flags option. TRUE: clear FALSE: not clear wait_option: WAIT_FOR_ANY_EVENT/ WAIT_FOR_ALL_EVENTS timeout_ms: Wait time before returning.

<b>Return</b>	<p>Result:</p> <ul style="list-style-type: none"> <li>• kNoErr: Indicates the successful completion of the service.</li> <li>• kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	<pre>uint32_t flags_set = 0; OSStatus result = lega_rtos_wait_for_event_flags(&amp;mEventFlags,     0x00ffffff, &amp;flags_set, TRUE, WAIT_FOR_ANY_EVENT,     LEGA_WAIT_FOREVER); if (flags_set &amp; 0x1) {     HandlePostEvent(); }</pre>

### 9.2.3 lega\_rtos\_set\_event\_flags

<b>Function</b>	This service sets event flags in an event flags group.
<b>Definition/Command</b>	OSStatus lega_rtos_set_event_flags(lega_event_flags_t *event_flags, uint32_t flags_to_set)
<b>Parameters</b>	event_flags: A pointer to the event_flags handler. flags_to_set: Event flags bits to set
<b>Return</b>	<p>Result:</p> <ul style="list-style-type: none"> <li>• kNoErr: Indicates the successful completion of the service.</li> <li>• kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	<pre>uint32_t flag_bits = 0x1; lega_rtos_set_event_flags(&amp;mEventFlags, flag_bits)</pre>

### 9.2.4 lega\_rtos\_deinit\_event\_flags

<b>Function</b>	This service deletes the specified event flags group.
<b>Definition/Command</b>	OSStatus lega_rtos_deinit_event_flags(lega_event_flags_t *event_flags)
<b>Parameters</b>	event_flags: A pointer to the event group handler
<b>Return</b>	<p>Result:</p> <ul style="list-style-type: none"> <li>• kNoErr: Indicates the successful completion of the service.</li> <li>• kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	

# 10.

# Utility

## 10.1 Functional Description

LEGA RTOS utility functions.

## 10.2 APIs

### 10.2.1 lega\_rtos\_get\_system\_version

<b>Function</b>	Gets system version strings.
<b>Definition/Command</b>	const char * lega_rtos_get_system_version()
<b>Parameters</b>	None
<b>Return</b>	<ul style="list-style-type: none"> <li>This service returns a pointer to a version string.</li> </ul>
<b>Notes</b>	
<b>Example</b>	printf("System version:%s", lega_rtos_get_system_version());

### 10.2.2 lega\_rtos\_task\_cfg\_get

<b>Function</b>	This api gets task configurations by indexes from the task_cfg table for convenient.(priority and stack size configuration), task_cfg table should be defined first.
<b>Definition/Command</b>	OSStatus lega_rtos_task_cfg_get(uint32_t index, lega_task_config_t *cfg)
<b>Parameters</b>	index: Task index in the cfg table cfg: A pointer of the config data structure of the task
<b>Return</b>	Result: <ul style="list-style-type: none"> <li>kNoErr: Indicates the successful completion of the service.</li> <li>kGeneralErr: Indicates an error occurring.</li> </ul>
<b>Notes</b>	
<b>Example</b>	<pre>lega_task_config_t task_cfg[LEGA_TASK_CONFIG_MAX] = {     {3, 1024}, // index LEGA_TASK_CONFIG_CHIP }; OSStatus lega_rtos_get_chip_task_cfg(lega_task_config_t *cfg) {     return lega_rtos_task_cfg_get(LEGA_TASK_CONFIG_CHIP, cfg); }</pre>



# 11. Memory

## 11.1 Functional Description

LEGA RTOS memory management functions.

## 11.2 APIs

### 11.2.1 lega\_rtos\_malloc

<b>Function</b>	malloc a buffer.
<b>Definition/Command</b>	void * lega_rtos_malloc(uint32_t xWantedSize)
<b>Parameters</b>	<b>xWantedSize:</b> wanted buffer size in bytes.
<b>Return</b>	<ul style="list-style-type: none"> <li>This service returns a pointer to buffer on success, NULL on fail.</li> </ul>
<b>Notes</b>	
<b>Example</b>	<pre>lega_mutex_t *m; m = lega_rtos_malloc(sizeof(lega_mutex_t)); if (m == NULL) {     return NULL; }</pre>

### 11.2.2 lega\_rtos\_free

<b>Function</b>	Free a buffer memory
<b>Definition/Command</b>	void lega_rtos_free(void *pv)
<b>Parameters</b>	<b>pv:</b> pointer to the buffer
<b>Return</b>	None
<b>Notes</b>	
<b>Example</b>	<pre>lega_mutex_t *m; m = lega_rtos_malloc(sizeof(lega_mutex_t)); if (m == NULL) {     return NULL; } if (lega_rtos_init_mutex(m) != kNoErr) {     lega_rtos_free(m);     return NULL; }</pre>