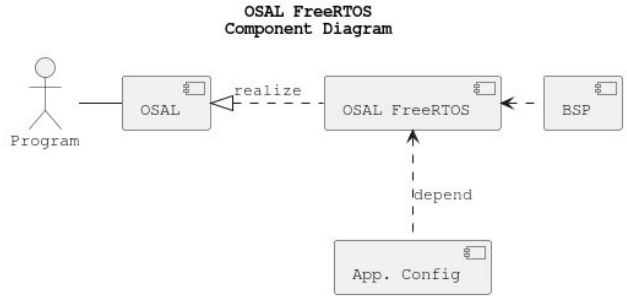
FreeRTOS

The OS variant is developed with **FreeRTOS v11.0 kernel**. The OS features a CMake and GCC build system. It requires an application and Board Support Package (BSP) configuration file that allows the user to set the CPU and peripheral clock frequency.

This OS variant was implemented on the basis of our generic and reusable Operating System Abstraction Layer (OSAL). This architecture ensures the portability of the operating system and minimizes the integration risks and the effort required for the transition to another operating system variant.

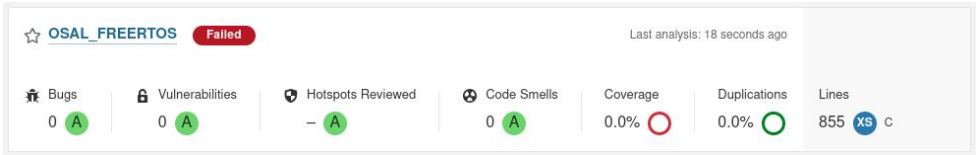
Some BSP peripherals such as CAN, DAC, I2C, NVM, PWM, SPI and UART offer a resource lock interface for thread-safe operation that already uses the OSAL definition.

Version: v0.1.0



Available Components

- ❑ CLOCK
- ❑ EVENT
- ❑ MEMORY
- ❑ QUEUE
- ❑ SEMAPHORE
- ❑ SLEEP
- ❑ THREAD
- ❑ TIMER



Clock APIs

```
osal_result_t osal_clock_get_ticks( uint32_t* )
osal_result_t osal_clock_get_time( osal_clock_time_t* )
osal_result_t osal_clock_set_time( const osal_clock_time_t )
```

Event APIs

```
osal_result_t osal_event_create( osal_event_handle_t* handle )
osal_result_t osal_event_delete( osal_event_handle_t* handle )
osal_result_t osal_event_set_bits( osal_event_handle_t*, uint32_t )
osal_result_t osal_event_get_bits( osal_event_handle_t*, uint32_t* )
Osal_result_t osal_event_wait_bits( osal_event_handle_t*, uint32_t, bool,
                                    bool, uint32_t, uint32_t* )
osal_result_t osal_event_clear_bits( osal_event_handle_t*, uint32_t )
```

Kernel APIs

```
osal_result_t osal_start( void )
osal_result_t osal_stop( void )
osal_result_t osal_suspend_all( void )
osal_result_t osal_resume_all( void )
```

Memory APIs

```
osal_result_t osal_memory_allocate( size_t, void** )
osal_result_t osal_memory_free( void* )
osal_result_t osal_memory_get_heap_info( osal_memory_heap_info_t* )
```

Timer APIs

```
osal_result_t osal_timer_create( char*, uint32_t, bool, void*,
                                 osal_timer_handle_t* )
osal_result_t osal_timer_delete( osal_timer_handle_t*, uint32_t )
osal_result_t osal_timer_reset( osal_timer_handle_t*, uint32_t )
osal_result_t osal_timer_start( osal_timer_handle_t*, uint32_t )
osal_result_t osal_timer_stop( osal_timer_handle_t*, uint32_t )
osal_result_t osal_timer_set_period( osal_timer_handle_t*, uint32_t, uint32_t )
osal_result_t osal_timer_get_remaining_time( osal_timer_handle_t*, uint32_t* )
```

Queue APIs

```
osal_result_t osal_queue_create( char*, size_t, size_t, osal_queue_handle_t* )
osal_result_t osal_queue_delete( osal_queue_handle_t* )
osal_result_t osal_queue_reset( osal_queue_handle_t* )
osal_result_t osal_queue_receive( osal_queue_handle_t*, void*, size_t*, uint32_t )
osal_result_t osal_queue_send( osal_queue_handle_t*, void*, size_t, uint32_t )
```

Semaphore APIs

```
osal_result_t osal_semphr_create( osal_semphr_type_t, osal_semphr_count_attr_t*,
                                  osal_semphr_handle_t* )
osal_result_t osal_semphr_delete( osal_semphr_handle_t* )
osal_result_t osal_semphr_take( osal_semphr_handle_t*, uint32_t )
osal_result_t osal_semphr_give( osal_semphr_handle_t* )
```

Sleep APIs

```
osal_result_t osal_sleep_sec( uint32_t )
osal_result_t osal_sleep_ms( uint32_t )
osal_result_t osal_sleep_us( uint32_t )
```

Thread APIs

```
osal_result_t osal_thread_create( char*, osal_thread_attr_t, osal_thread_fn,
                                  void*, osal_thread_handle_t* )
osal_result_t osal_thread_delete( osal_thread_handle_t* )
osal_result_t osal_thread_suspend( osal_thread_handle_t* )
osal_result_t osal_thread_resume( osal_thread_handle_t* )
osal_result_t osal_thread_get_handle( osal_thread_handle_t* )
osal_result_t osal_thread_get_info( osal_thread_handle_t*, osal_thread_info_t* )
osal_result_t osal_thread_delay( uint32_t )
```