Checking If FreeRTOS Task is Blocked on Synchronization Primitives

In FreeRTOS, determining if a task is blocked on a synchronization primitive and identifying which specific object it's waiting for requires understanding both the task's current state and the mechanisms available for runtime inspection.

Understanding Task States and Blocking

When a FreeRTOS task calls a blocking function like `xSemaphoreTake()`, `xQueueReceive()`, or `xEventGroupWaitBits()`, the task transitions from the Running or Ready state to the Blocked state. The scheduler maintains this state information implicitly by placing tasks in appropriate lists rather than storing a state variable in each task.[1] [2]

Tasks are reported with the following state indicators when using debugging functions:[3] [4]

- **'B'** - Blocked (waiting on a synchronization object)
- **'R'** - Ready to run
- **'X'** - Currently running/executing[5]
- **'S'** - Suspended
- **'D'** - Deleted

Primary Methods for Inspection

1. Using `eTaskGetState()` for Basic State Checking

The `eTaskGetState()` function provides the current state of any task:[6]

```
#include "FreeRTOS.h"
#include "task.h"

eTaskState taskState = eTaskGetState(taskHandle);
if (taskState == eBlocked) {
    // Task is blocked on some synchronization primitive
}
```

Requirements: Set `INCLUDE_eTaskGetState` to 1 in `FreeRTOSConfig.h`.[7]

2. Using `uxTaskGetSystemState()` for Detailed Information

This function provides comprehensive task information including what object a task is blocked on:[8] [9]

```
TaskStatus_t taskStatusArray[^1_10]; // Adjust size as needed
UBaseType_t taskCount;
uint32_t totalRunTime;

taskCount = uxTaskGetSystemState(taskStatusArray,
                                 sizeof(taskStatusArray) / sizeof(TaskStatus_t),
                                 &totalRunTime);

for (int i = 0; i < taskCount; i++) {
    if (taskStatusArray[i].eCurrentState == eBlocked) {
        // Task is blocked - check what it's waiting on
        printf("Task %s is blocked\n", taskStatusArray[i].pcTaskName);
    }
}
```

Requirements: Set `configUSE_TRACE_FACILITY` to 1 in `FreeRTOSConfig.h`.[8]

3. Using `vTaskList()` for Human-Readable Output

The `vTaskList()` function provides formatted task information including blocking objects:[10] [3]

```
char taskListBuffer[^1_1024]; // 40 bytes per task minimum
vTaskList(taskListBuffer);
printf("%s", taskListBuffer);
```

Requirements: Both `configUSE_TRACE_FACILITY` and `configUSE_STATS_FORMATTING_FUNCTIONS` must be set to 1.[4] [3]

4. Using `vTaskGetInfo()` for Single Task Details

For detailed information about a specific task:[11] [12]

```
TaskStatus_t taskStatus;
vTaskGetInfo(taskHandle, &taskStatus, pdTRUE, eInvalid);

if (taskStatus.eCurrentState == eBlocked) {
    // Task is blocked - additional details available in taskStatus
}
```

Identifying the Specific Blocking Object

Queue Registry for Named Objects

To identify which specific synchronization object is blocking a task, use the queue registry to assign human-readable names to your synchronization primitives:[13]

```
// During initialization
xSemaphoreHandle mySemaphore = xSemaphoreCreateBinary();
vQueueAddToRegistry(mySemaphore, "MySemaphore");

xQueueHandle myQueue = xQueueCreate(10, sizeof(int));
vQueueAddToRegistry(myQueue, "MyQueue");

EventGroupHandle_t myEventGroup = xEventGroupCreate();
vQueueAddToRegistry(myEventGroup, "MyEventGroup");
```

When using debugging tools or `vTaskList()`, blocked tasks will show the registered name of the object they're waiting on instead of just a memory address.[14] [13]

TaskStatus_t Structure Information

The `TaskStatus_t` structure contains several relevant members:[15]

```
typedef struct {
    TaskHandle_t xHandle;
    const char *pcTaskName;
    UBaseType_t xTaskNumber;
    eTaskState eCurrentState;
    UBaseType_t uxCurrentPriority;
    UBaseType_t uxBasePriority;
    uint32_t ulRunTimeCounter;
    StackType_t *pxStackBase;
    uint16_t usStackHighWaterMark;
} TaskStatus_t;
```

Practical Implementation Example

Here's a comprehensive function to check if a task is blocked and attempt to identify the blocking object:

```
void checkTaskBlockingStatus(TaskHandle_t taskHandle, const char* taskName) {
    eTaskState state = eTaskGetState(taskHandle);

    if (state == eBlocked) {
        printf("Task '%s' is BLOCKED on a synchronization primitive\n", taskName);

        // Get detailed information
        TaskStatus_t taskStatus;
        vTaskGetInfo(taskHandle, &taskStatus, pdFALSE, eInvalid);
```

```
        // For more detailed debugging, use vTaskList
        char taskListBuffer[^1_1024];
        vTaskList(taskListBuffer);

        // Parse the output to find the specific task and its event object
        // The event object will be shown if registered with vQueueAddToRegistry
        printf("Full task list:\n%s\n", taskListBuffer);

    } else {
        printf("Task '%s' is not blocked (state: %d)\n", taskName, state);
    }
}
```

Debugging Considerations

Performance Impact: Functions like `uxTaskGetSystemState()` and `vTaskList()` disable interrupts for their duration and are intended primarily for debugging.[9] [3]

State Volatility: Task states can change between the time you query them and when you act on the information.[7]

Stack Analysis: For more advanced debugging, you can examine the task's stack to determine the call chain that led to the blocking state, though this requires deeper knowledge of the specific blocking function implementations.[16]

IDE Integration: Many development environments provide FreeRTOS-aware debugging plugins that can automatically display which synchronization objects are blocking each task.[17] [14]

By combining these techniques—particularly using `eTaskGetState()` for quick state checking, the queue registry for object identification, and `vTaskList()` or `uxTaskGetSystemState()` for comprehensive analysis—you can effectively determine both whether a task is blocked and identify the specific synchronization primitive causing the blocking condition.

⁂

Debugging FreeRTOS Task States Using Only a Debugger

When debugging FreeRTOS with only a debugger, you need to understand the internal data structures and how tasks are organized within the kernel's memory. Here's how to inspect task blocking states and the underlying "appropriate lists" using debugger variables and memory inspection.

Understanding FreeRTOS Internal Task Lists

FreeRTOS tracks task states implicitly by placing tasks in specific lists rather than storing explicit state variables in each task. The "appropriate lists" mentioned are internal linked lists maintained by the kernel:[69]

Key Global List Variables

The following global variables in `tasks.c` contain the various task lists that you can inspect in your debugger:[70] [71]

Ready Task Lists:

- `pxReadyTasksLists[]` - Array of lists, one for each priority level containing ready tasks[72] [69]
- `pxCurrentTCB` - Pointer to the currently executing task's TCB[73] [74]

Blocked Task Lists:

- `pxDelayedTaskList` - Pointer to list of tasks blocked with timeout (e.g., `vTaskDelay`)[75] [76] [77]
- `pxOverflowDelayedTaskList` - Overflow list when delay values wrap around[78] [71]
- `xSuspendedTaskList` - List of suspended tasks[69]
- `xPendingReadyList` - Tasks that became ready while scheduler was suspended[71]

Additional Lists:

- `xTasksWaitingTermination` - Tasks waiting to be cleaned up after deletion[71]

Task Control Block (TCB) Structure

Each task has a Task Control Block containing two critical list items that you can inspect in the debugger:[73] [69]

```
typedef struct tskTaskControlBlock {
    volatile StackType_t *pxTopOfStack;        // Current stack pointer
    ListItem_t xGenericListItem;               // For ready/blocked lists
    ListItem_t xEventListItem;                 // For synchronization objects
    UBaseType_t uxPriority;
    StackType_t *pxStack;
    char pcTaskName[configMAX_TASK_NAME_LEN];
    // ... other members
} TCB_t;
```

Debugger Inspection Techniques

1. Check Current Task State via TCB Location

To determine if a task is blocked and on which object, inspect the `pxCurrentTCB` variable:[74]

In your debugger variables window:

```
pxCurrentTCB                    // Current task pointer
(tskTCB*)pxCurrentTCB           // Cast to see TCB structure
pxCurrentTCB->pcTaskName        // Task name
pxCurrentTCB->xEventListItem    // Event list information
```

2. Inspect Event List Item for Blocking Object

The key to identifying what a task is blocked on lies in the `xEventListItem` member of the TCB: [79] [80]

Check these TCB members:

```
pxCurrentTCB->xEventListItem.pvContainer    // Points to the synchronization object
pxCurrentTCB->xEventListItem.xItemValue     // Timeout or priority value
```

Critical insight: If `pvContainer` is **not NULL**, the task is blocked on a synchronization primitive. The `pvContainer` points to the queue, semaphore, or event group the task is waiting on.[80]

3. Determine Which List Contains the Task

You can manually inspect the task lists to see where your task is located:

For blocked tasks:

```
pxDelayedTaskList               // Check if task is in delayed list
pxOverflowDelayedTaskList       // Check overflow delayed list
xSuspendedTaskList              // Check suspended list
```

For ready tasks:

```
pxReadyTasksLists[^2_0]            // Priority 0 ready tasks
pxReadyTasksLists[^2_1]            // Priority 1 ready tasks
// ... continue for each priority level
```

4. Inspecting Synchronization Objects

When you find a task blocked on a synchronization object via `pvContainer`, cast it to the appropriate type:[81]

For queues/semaphores:

```
(Queue_t*)pvContainer->pcTail           // Queue structure
(Queue_t*)pvContainer->pcHead           // Queue head pointer
(Queue_t*)pvContainer->uxMessagesWaiting // Messages in queue
```

For event groups:

```
(EventGroup_t*)pvContainer->uxEventBits  // Current event bits
```

Debugger-Specific Inspection Methods

Memory Window Analysis

Use your debugger's memory window to examine the list structures directly:[73]

1. **Find the task handle** (which is actually a pointer to the TCB)
2. **Navigate to the TCB structure** in memory
3. **Examine the** `xEventListItem.pvContainer` field
4. **If non-NULL, inspect the memory at that address** to see the synchronization object

Watch Expressions

Set up watch expressions in your debugger to monitor key variables:[82]

```
pxCurrentTCB->pcTaskName                    // Current task name
pxCurrentTCB->xEventListItem.pvContainer    // Blocking object pointer
pxDelayedTaskList->uxNumberOfItems          // Number of delayed tasks
pxReadyTasksLists[priority].uxNumberOfItems // Ready tasks at priority
```

Debugging Symbols Required

For this approach to work, ensure your project has debugging symbols enabled and the following FreeRTOS internal symbols are accessible:[71]

- `pxCurrentTCB`
- `pxReadyTasksLists`
- `pxDelayedTaskList`
- `pxOverflowDelayedTaskList`
- `xSuspendedTaskList`

Practical Debugging Workflow

1. **Pause execution** when you suspect a task is blocked
2. **Check** `pxCurrentTCB->pcTaskName` to identify the current task
3. **Examine** `pxCurrentTCB->xEventListItem.pvContainer`:
   - If **NULL**: Task is not blocked on a synchronization object (may be in ready list or delayed)
   - If **non-NULL**: Task is blocked on the synchronization object at this address
4. **Cast the** `pvContainer` **address** to the appropriate synchronization object type
5. **Inspect the object's properties** to understand why the task is blocked

This approach gives you complete visibility into FreeRTOS task states and blocking conditions using only standard debugger memory and variable inspection capabilities, without requiring

specialized RTOS-aware debugging plugins.[81] [82]

⁂

1. https://stackoverflow.com/questions/41254139/how-scheduler-knows-a-task-is-in-blocking-state
2. https://aosabook.org/en/v2/freertos.html
3. https://docs.espressif.com/projects/esp-idf/en/v4.3/esp32/api-reference/system/freertos.html
4. https://sourcevu.sysprogs.com/stm32/Libraries/FreeRTOS/symbols/vTaskList
5. https://www.reddit.com/r/embedded/comments/1as36bk/what_is_the_meaning_of_x_of_task_status_printed/
6. https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/system/freertos_idf.html
7. https://sourcevu.sysprogs.com/rp2040/freertos/symbols/eTaskGetState
8. https://docs.espressif.com/projects/esp-idf/en/release-v3.1/api-reference/system/freertos.html
9. https://freertos.org/uxTaskGetSystemState.html
10. https://www.instructables.com/FreeRTOS-With-Arduino-09-Read-Task-Info-VTaskList/
11. https://sourcevu.sysprogs.com/rp2040/freertos/symbols/vTaskGetInfo
12. https://www.freertos.org/Documentation/02-Kernel/04-API-references/03-Task-utilities/02-vTaskGetInfo
13. https://community.st.com/t5/stm32cubeide-mcus/unable-to-locate-the-os-resource-that-is-blocking-the-task/td-p/129560
14. https://www.freertos.org/FreeRTOS_Support_Forum_Archive/April_2010/freertos_How_to_know_where_my_tasks_are_waiting_3686456.html
15. https://docs.espressif.com/projects/esp-idf/en/release-v3.0/api-reference/system/freertos.html
16. https://stackoverflow.com/questions/44709336/get-info-about-what-resource-is-blocking-task-in-freertos
17. https://m.freertos.org/FreeRTOS_Support_Forum_Archive/September_2014/freertos_Blocked_Task_..._b2cd6ed4j.html
18. https://stackoverflow.com/questions/69027678/how-to-know-which-task-has-taken-a-binary-semaphore
19. https://www.freertos.org/FreeRTOS_Support_Forum_Archive/January_2019/freertos_Debugging_suspended_tasks_22414aeadaj.html
20. https://forums.freertos.org/t/task-blocking-on-various-types-of-primitives/12394
21. https://www.reddit.com/r/embedded/comments/yky5s2/freertos_should_i_use_events_or_a_binary_semaphore/
22. https://www.youtube.com/watch?v=So_O4GiqWIA
23. https://docs.aws.amazon.com/freertos/latest/userguide/inter-task-coordination.html
24. https://freertos.org/FreeRTOS_Support_Forum_Archive/October_2012/freertos_Unexpected_task_behav_when_blocked_on_a_queue_5905842.html
25. https://freertos.org/Documentation/02-Kernel/02-Kernel-features/01-Tasks-and-co-routines/04-Task-scheduling
26. https://forums.freertos.org/t/how-can-i-find-out-if-another-task-is-currently-blocked-waiting-for-my-binary-semaphore/16462?page=2

27. https://www.freertos.org/Documentation/02-Kernel/04-API-references/03-Task-utilities/10-xTaskCheckForTimeOut
28. https://software-dl.ti.com/simplelink/esd/simplelink_cc13xx_cc26xx_sdk/8.30.01.01/exports/docs/ble5stack/ble_user_guide/html/freertos/synchronization.html
29. https://freertos.org/RTOS_Task_Notification_As_Counting_Semaphore.html
30. https://forums.freertos.org/t/how-do-i-check-on-the-state-of-a-task-how-do-i-know-when-it-has-been-deleted/11996
31. https://stackoverflow.com/questions/44894205/freertos-vs-zephyr-mynewt-task-blocked-state
32. https://www.freertos.org/Documentation/02-Kernel/02-Kernel-features/03-Direct-to-task-notifications/01-Task-notifications
33. https://www.freertos.org/Documentation/02-Kernel/04-API-references/03-Task-utilities/09-vTaskSetTimeOutState
34. https://www.freertos.org/FreeRTOS_Support_Forum_Archive/June_2019/freertos_Blocking_on_Multiple_RTOS_Objects_ecf83b54e4j.html
35. https://community.st.com/t5/stm32-mcus-embedded-software/freertos-waiting-flag-in-task/td-p/369052
36. https://forums.freertos.org/t/how-do-i-get-all-task-handles-without-calling-uxtaskgetsystemstate/16451
37. https://www.freertos.org/FreeRTOS_Support_Forum_Archive/June_2012/freertos_Enhance_vTaskList_and_vTaskGetRunTimeStats_5318855.html
38. http://www.openrtos.net/a00021.html
39. http://www.openrtos.net/uxTaskGetSystemState.html
40. https://esp32.com/viewtopic.php?t=5648
41. https://esp32.com/viewtopic.php?t=3674
42. https://www.freertos.org/Documentation/02-Kernel/04-API-references/03-Task-utilities/00-Task-utilities
43. https://esp32.com/viewtopic.php?t=39278
44. https://freertos.org/zh-cn-cmn-s/Documentation/02-Kernel/04-API-references/03-Task-utilities/01-uxTaskGetSystemState
45. https://github.com/espressif/arduino-esp32/issues/2203
46. https://github.com/espressif/arduino-esp32/issues/7179
47. https://www.reddit.com/r/embedded/comments/189emuv/freertos_task_blocked_on_stm32/
48. https://www.reddit.com/r/embedded/comments/10i2gb2/how_can_i_use_event_flags_in_freertos_properly/
49. https://embeddedexplorer.com/task-synchronization-with-freertos-event-groups-on-esp32-esp-wrover-kit-example/
50. https://stackoverflow.com/questions/35620365/freertos-blocking-on-multiple-events-objects
51. https://www.freertos.org/FreeRTOS_Support_Forum_Archive/March_2017/freertos_give_semaphore_only_if_task_s_is_blocked_by_the_semaphore_f63cd859j.html
52. https://labs.dese.iisc.ac.in/embeddedlab/freertos-task-notifications/
53. https://forums.freertos.org/t/block-task-waiting-for-flag-change/1024
54. https://software-dl.ti.com/simplelink/esd/simplelink_cc13xx_cc26xx_sdk/7.40.00.77/exports/docs/ble5stack/ble_user_guide/html/freertos/synchronization.html

55. https://community.nxp.com/t5/Kinetis-Microcontrollers/FreeRTOS-Task-Overflow-Detection/m-p/1262822/?profile.language=en
56. https://freertos.org/Documentation/02-Kernel/02-Kernel-features/02-Queues-mutexes-and-semaphores/01-Queues
57. https://stackoverflow.com/questions/70260081/freertos-queue-vs-semaphore
58. https://stackoverflow.com/questions/50488701/how-to-make-many-freertos-tasks-wait-for-one-other-to-complete-initialization
59. https://www.freertos.org/Documentation/02-Kernel/02-Kernel-features/10-Blocking-on-multiple-RTOS-objects
60. https://forums.freertos.org/t/i-need-help-with-task-synchronization-in-freertos/20737
61. https://www.freertos.org/Documentation/02-Kernel/04-API-references/12-Event-groups-or-flags/11-xEventGroupSync
62. https://stackoverflow.com/questions/59508515/rtos-tcb-vs-task-stack
63. https://people.montefiore.uliege.be/boigelot/cours/embedded/slides/embedded-ch-6.pdf
64. https://www.freertos.org/media/2018/FreeRTOS_Reference_Manual_V10.0.0.pdf
65. https://stackoverflow.com/questions/62794291/how-to-debug-a-freertos-application
66. https://circuitcellar.com/research-design-hub/design-solutions/freertos-part-2/
67. https://www.freertos.org/media/2025/FreeRTOS_Reference_Manual_V8.2.1.pdf
68. https://www.reddit.com/r/embedded/comments/1d50o47/freertos_tasknotification_vs_eventgroups_whats/
69. https://aosabook.org/en/v2/freertos.html
70. https://forums.freertos.org/t/how-tcbs-are-managed-in-freertos/10435
71. https://github.com/Marus/cortex-debug/issues/314
72. https://sourcevu.sysprogs.com/espressif/lib/freertos/symbols/pxReadyTasksLists
73. https://tewarid.github.io/2014/03/25/freertos-task-control-block-and-stack-in-avr32.html
74. https://www.freertos.org/FreeRTOS_Support_Forum_Archive/October_2011/freertos_Debug_a_function_which_used_by_several_Tasks_4757655.html
75. https://forums.freertos.org/t/issue-with-xtaskdelayuntil-api/19868
76. https://freertos.org/FreeRTOS_Support_Forum_Archive/April_2013/freertos_Delayed_task_lists_crashes_7750649.html
77. https://forums.freertos.org/t/freertos-null-pointer-in-xtaskincrementtick/12060
78. https://stackoverflow.com/questions/58744284/how-to-run-a-patch-file-with-several-diffs-inside
79. https://community.nxp.com/t5/MCUXpresso-IDE/Freertos-debug-issue/m-p/1403362
80. https://forums.freertos.org/t/debugging-suspended-tasks/7642
81. https://www.reddit.com/r/embedded/comments/189emuv/freertos_task_blocked_on_stm32/
82. http://bikealive.nl/debugging-freertos.html
83. https://www.freertos.org/FreeRTOS_Support_Forum_Archive/January_2019/freertos_Debugging_suspended_tasks_22414aeadaj.html
84. https://www.youtube.com/watch?v=So_O4GiqWIA
85. https://circuitcellar.com/research-design-hub/design-solutions/freertos-part-2/

86. https://www.freertos.org/FreeRTOS_Support_Forum_Archive/September_2015/freertos_MSP430X_Application_Stuck_in_Endless_Loop_Inside_xTaskIncrementTick_FreeRTOS_v8.2.2_1d196f43j.html
87. https://stackoverflow.com/questions/62794291/how-to-debug-a-freertos-application
88. https://stackoverflow.com/questions/59508515/rtos-tcb-vs-task-stack
89. https://forums.freertos.org/t/getting-all-the-tasks-via-the-debugger/8143
90. https://www.freertos.org/FreeRTOS_Support_Forum_Archive/October_2017/freertos_how_can_I_get_the_TCB_imformation_from_the_task_handle_3232a35ej.html
91. https://sol.sbc.org.br/index.php/wperformance/article/download/15728/15569/
92. https://community.st.com/t5/stm32-mcus/using-vs-code-for-freertos-application-debugging/ta-p/756316
93. https://homel.vsb.cz/~sta048/mcu/doc/freertos/Mastering-the-FreeRTOS-Real-Time-Kernel.v1.0.pdf
94. https://mcuoneclipse.com/2017/07/27/troubleshooting-tips-for-freertos-thread-aware-debugging-in-eclipse/
95. https://community.st.com/t5/stm32-mcus/how-to-enable-freertos-run-time-and-stack-usage-view/ta-p/627524
96. https://www.freertos.org/FreeRTOS_Support_Forum_Archive/October_2013/freertos_vTaskIncrementTick_changed_to_xTaskIncrementTick_2dcb1205j.html
97. https://community.st.com/t5/stm32cubemx-mcus/malloc-returns-null-pointer-in-freertos-task/td-p/333558
98. https://community.st.com/t5/stm32-mcus-embedded-software/how-to-debug-freertos/td-p/434884
99. https://stackoverflow.com/questions/56703232/how-to-show-runtime-in-freertos-task-list-during-debugging
100. https://freertos.org/FreeRTOS_Support_Forum_Archive/August_2017/freertos_xQueueGenericReceive_Queue_Address_corruption_716272d2j.html
101. https://stackoverflow.com/questions/36590893/freertos-debugging-with-openocd
102. https://www.freertos.org/Documentation/02-Kernel/03-Supported-devices/02-Customization
103. https://sysprogs.com/w/forums/topic/unable-to-debug-a-freertos-project/
104. https://community.nxp.com/t5/-/-/m-p/755094
105. https://freertos.org/Documentation/02-Kernel/04-API-references/01-Task-creation/02-xTaskCreateStatic
106. https://community.platformio.org/t/debugging-stm32f407-freertos-cant-break-point-other-threads/31618
107. https://community.st.com/t5/stm32cubeide-mcus/freertos-debug-windows-not-displaying-proper-data/td-p/142023
108. https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/system/freertos_idf.html
109. https://www.freertos.org/Documentation/02-Kernel/06-Coding-guidelines/01-Source-code-organization
110. https://community.nxp.com/t5/MCUXpresso-IDE/Freertos-debug-issue/m-p/1403837
111. https://www.freertos.org/Documentation/02-Kernel/02-Kernel-features/01-Tasks-and-co-routines/00-Tasks-and-co-routines
112. https://community.st.com/t5/stm32-mcus-embedded-software/rtos-debug-with-stm32cubeide/td-p/129486/page/3

113. https://www.freertos.org/Documentation/02-Kernel/02-Kernel-features/01-Tasks-and-co-routines/05-Implementing-a-task
114. https://community.nxp.com/t5/MCUXpresso-IDE/Freertos-debug-issue/m-p/1403312/1000
115. https://www.freertos.org/FreeRTOS_Support_Forum_Archive/December_2014/freertos_FreeRTOS_uxNumberOfItems_of_pxReadyTasksLists_uxTopPriority_becomes_0_randomly_3b79175aj.html
116. https://freertos.org/FreeRTOS-Coding-Standard-and-Style-Guide.html
117. http://huntershuai.github.io/2014/04/25/FreeRTOSlearning.html
118. https://github.com/Marus/cortex-debug/issues/1007
119. https://community.st.com/ysqtg83639/attachments/ysqtg83639/stm32-mcu-cubeide-forum/26498/1/FreeRTOS.debugging.on.STM32.pdf