|  |
| --- |
| **Software Test Plan** |

|  |  |
| --- | --- |
| Nr. : | 01 |
| Title: | OS Task Manager |

**Contents**

1. Test Specification Information 3

2. Module Test Cases 3

3. Integration Test Cases 12

# Test Specification Information

|  |  |  |
| --- | --- | --- |
| **Date of issue (MM/DD/YY)** | **Test Developer** | **Revision & Description** |
| 25/03/2014 | Miguel, Sergio, Esteban | 2.0 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Module Test Cases

|  |  |  |
| --- | --- | --- |
| **Test Case** | **ID** | **Status** |
| File Structure | **2.0** | **Done** |
| **Requirements covered** | | |
| 2.0 | | |
| **Test Procedure** | | |
| Os folder will be provided in Services layer containing the  following files: Os\_TaskM.c/Os\_TaskM.h, Os\_TaskCfg.c/Os\_TaskCfg.h, Os\_Task.c/Os\_Task.h, Os\_Types.h | | |
| **Expected Results** | | |
| Os folder will be provided in Services layer. | | |
| **Actual Results** | | **Test Results** |
| Os folder is provided in Services layer. See Figure 1 | | PASS |
| **Comments** | | |
|  | | |

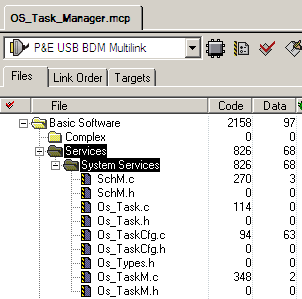
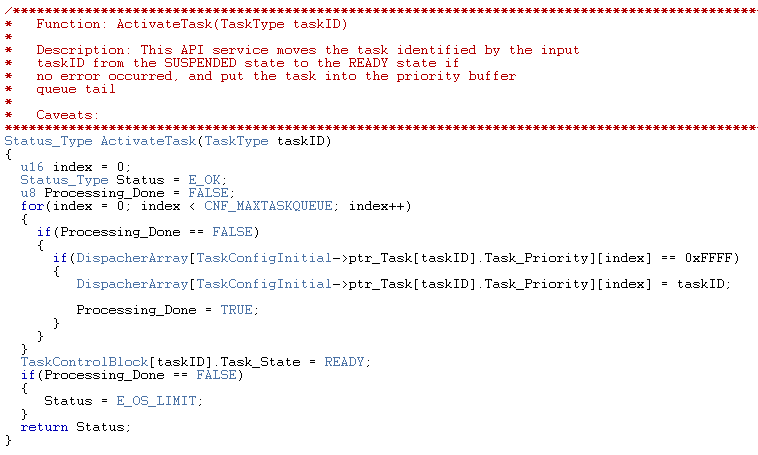
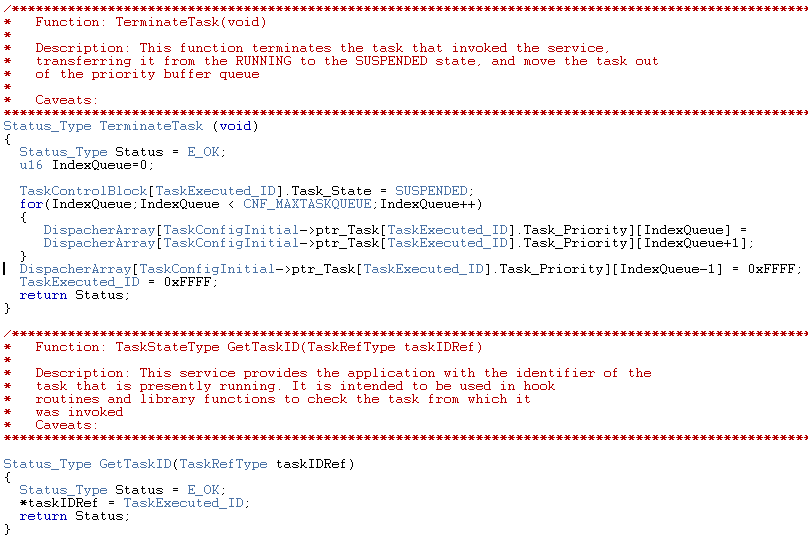


Figure 1

|  |  |  |
| --- | --- | --- |
| **Test Case** | **ID** | **Status** |
| Task Manager Services | **2.1** | **Done** |
| **Requirements covered** | | |
| 2.1 | | |
| **Test Procedure** | | |
| Os\_TaskM.c/Os\_TaskM.h: provide the following Task Manager Services: AvtivateTask, TerminateTask, GetTaskID, GetTaskState | | |
| **Expected Results** | | |
| Task Manager Services included in Os\_TaskM.c & Os\_TaskM.h | | |
| **Actual Results** | | **Test Results** |
| Task Manager Services are provided in Os\_TaskM.c & Os\_TaskM.h,  See Figure 2 & Figure 3 | | PASS |
| **Comments** | | |
|  | | |

Os\_TaskM.c





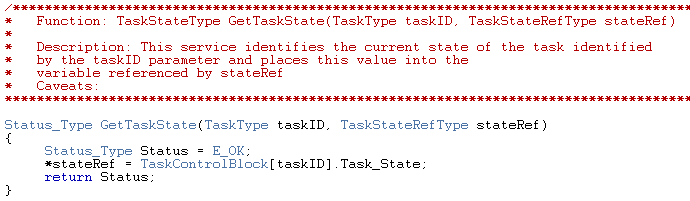


Figure 2

Os\_TaskM.h

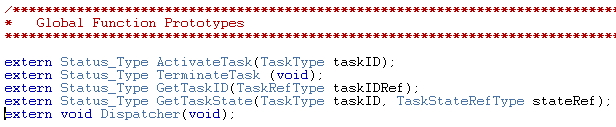


Figure 3

|  |  |  |
| --- | --- | --- |
| **Test Case** | **ID** | **Status** |
| Configuration Support | **2.2** | **Done** |
| **Requirements covered** | | |
| 2.2 | | |
| **Test Procedure** | | |
| Os\_TaskCfg.c/Os\_TaskCfg.h: provide OS tasks configuration support, these files will replace SchM\_Cfg.c/SchM\_Cfg.h which will not be provided | | |
| **Expected Results** | | |
| Os\_TaskCfg.c/Os\_TaskCfg.h: are used to provide OS tasks configuration support.  SchM\_Cfg.c/SchM\_Cfg.h not available | | |
| **Actual Results** | | **Test Results** |
| Os\_TaskCfg.c/Os\_TaskCfg.h: provide OS tasks configuration support.  SchM\_Cfg.c/SchM\_Cfg.h are not longer available. | | PASS |
| **Comments** | | |
| See Figure 4 & Figure 5. | | |

Os\_TaskCfg.h



Figure 4

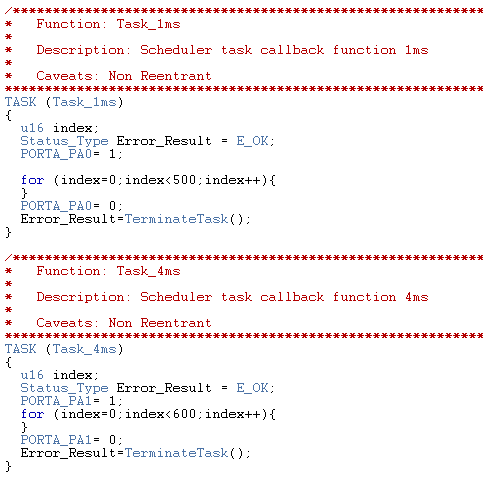
Os\_TaskCfg.c

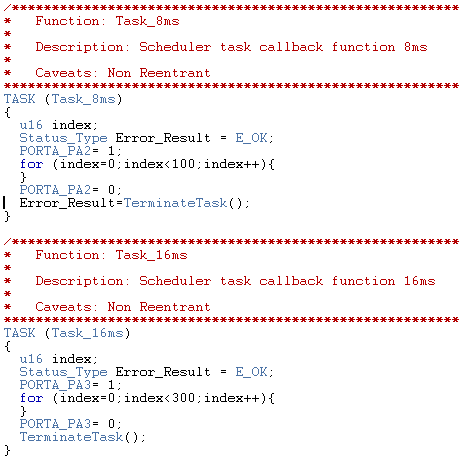


Figure 5

|  |  |  |
| --- | --- | --- |
| **Test Case** | **ID** | **Status** |
| Task Configuration | **2.3** | **Done** |
| **Requirements covered** | | |
| 2.3, 2.7, 2.9 | | |
| **Test Procedure** | | |
| Os\_Task.c/Os\_Task.h: provide the configured tasks to the application replacing SchM\_Task.c/SchM\_Task.h | | |
| **Expected Results** | | |
| Configured tasks are provided in Os\_Task.c & Os\_Task.h | | |
| **Actual Results** | | **Test Results** |
| SchM\_Task.c/SchM\_Task.h are not longer available and Os\_Task.c & Os\_Task.h provide tasks configuration. | | PASS |
| **Comments** | | |
| See Figure 4 & Figure 5 | | |

Os\_Task.c





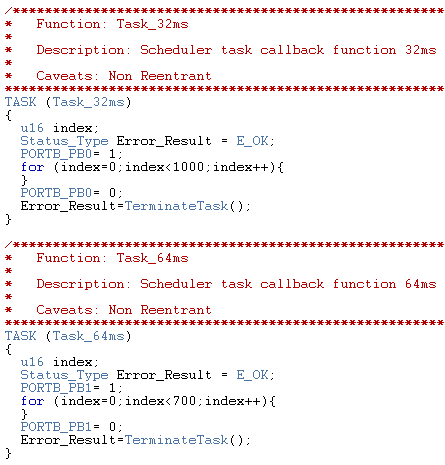


Figure 4

Os\_Task.h

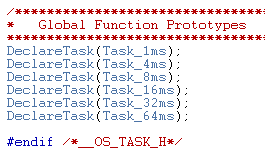


Figure 5

|  |  |  |
| --- | --- | --- |
| **Test Case** | **ID** | **Status** |
| Data definitions | **2.4** | **Done** |
| **Requirements covered** | | |
| 2.4, 2.8, 2.10 | | |
| **Test Procedure** | | |
| Provide the required data definitions for Task Management services support in Os\_Types.h and replace SchM\_Types.h | | |
| **Expected Results** | | |
| Data definitions for Task Management services support is provided in Os\_Types.h | | |
| **Actual Results** | | **Test Results** |
| Os\_Types.h is not longer available and all Data definitions for Task Management services are supported in Os\_Types.h | | PASS |
| **Comments** | | |
| See Figure 6 | | |

Os\_Types.h

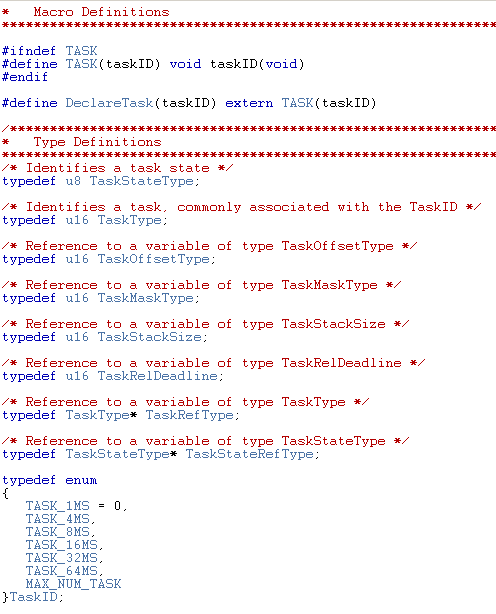
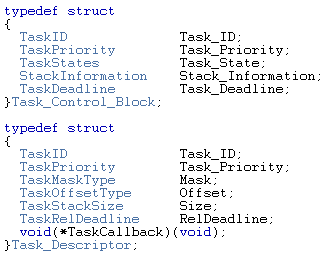
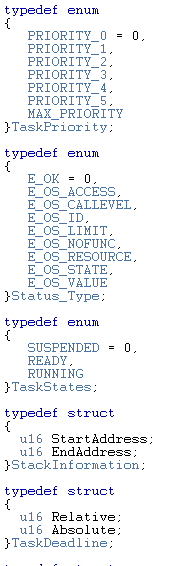
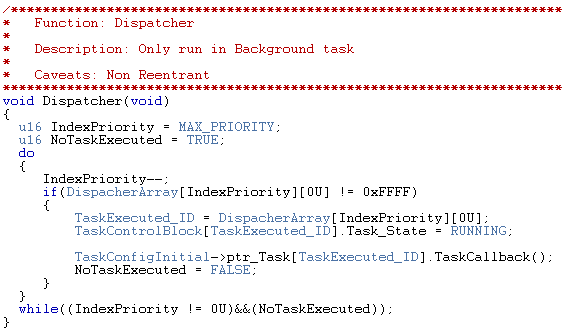
 

Figure 6

|  |  |  |
| --- | --- | --- |
| **Test Case** | **ID** | **Status** |
| Additional Information | **2.5** | **Done** |
| **Requirements covered** | | |
| 2.5, 2.6, 2.12 | | |
| **Test Procedure** | | |
| Define/declare Dispatcher in SchM.c/SchM.h files respectively & provide the Background task in SchM.c file which must call Dispatcher service. | | |
| **Expected Results** | | |
| Dispatcher declaration is inside of SchM.c & SchM.h files. Additionally Background task included only in SchM.c | | |
| **Actual Results** | | **Test Results** |
| Dispatcher is defined in SchM.c & SchM.h and Background task is included in SchM.c only. | | PASS |
| **Comments** | | |
| See Figure 7 & Figure 8 | | |

SchM.c



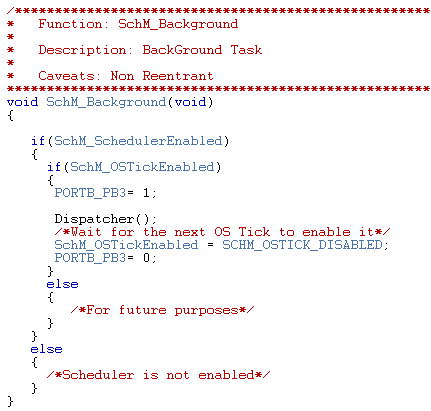


Figure 7

SchM.h

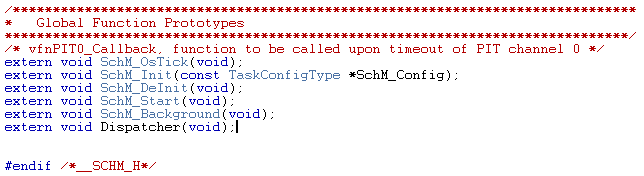


Figure 8

|  |  |  |
| --- | --- | --- |
| **Test Case** | **ID** | **Status** |
| Scheduler Services | **2.6** | **Done** |
| **Requirements covered** | | |
| 2.11 | | |
| **Test Procedure** | | |
| Execute scheduler algorithm and ActivateTask in SchM\_OsTick | | |
| **Expected Results** | | |
| Scheduler algorithm and ActivateTask have to be performed in SchM\_OsTick | | |
| **Actual Results** | | **Test Results** |
| Scheduler algorithm and ActivateTask are performed in SchM\_OsTick | | PASS |
| **Comments** | | |
| See Figure 9 | | |

SchM.c

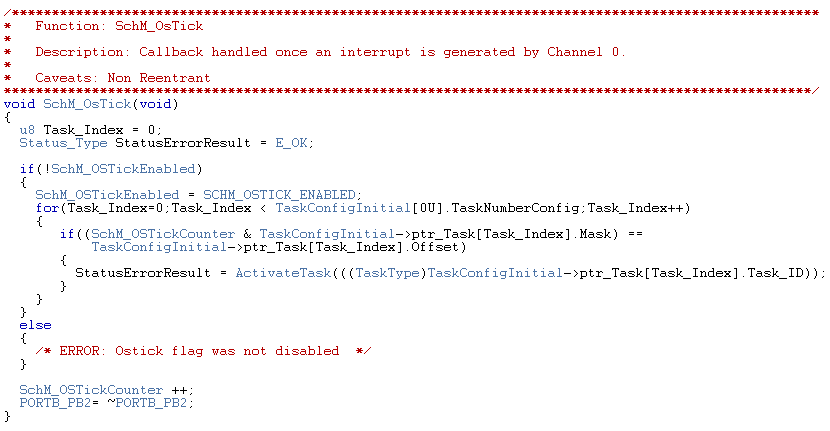


Figure 9

|  |  |  |
| --- | --- | --- |
| **Test Case** | **ID** | **Status** |
| Background | **2.7** | **Done** |
| **Requirements covered** | | |
| 2.13 | | |
| **Test Procedure** | | |
| Call SchM\_Start inside of main, then call SchM\_Background. | | |
| **Expected Results** | | |
| SchM\_Start must be called from main, after that SchM\_Background should be called and must never ends | | |
| **Actual Results** | | **Test Results** |
| SchM\_Background is called after SchM\_Start inside of main and never ends | | PASS |
| **Comments** | | |
| See Figure 13 & Figure 14 | | |

Main.c

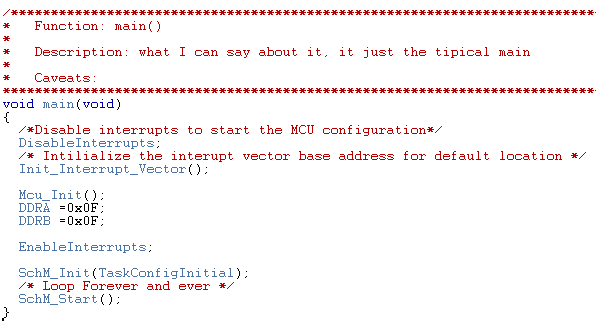


Figure 13

SchM.c

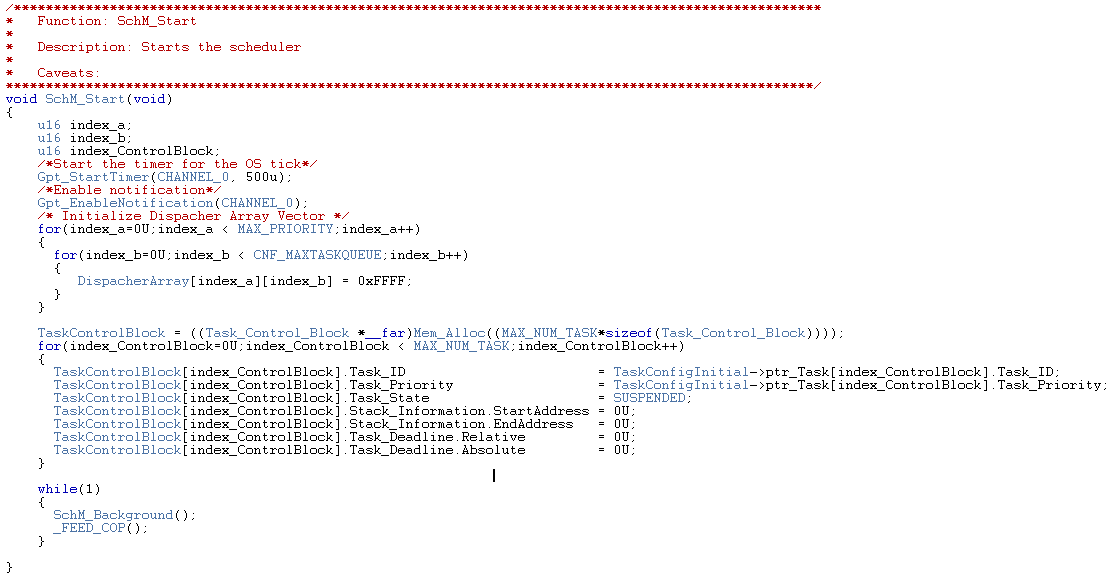


Figure 14

|  |  |  |
| --- | --- | --- |
| **Test Case** | **ID** | **Status** |
| Priority | **2.8** | **Done** |
| **Requirements covered** | | |
| 2.14, 2.15 | | |
| **Test Procedure** | | |
| Assign the lowest priority value to the lowest priority task, assign the same priority to a few tasks | | |
| **Expected Results** | | |
| Task execution can be performed according its priority assigned even if they have the same priority | | |
| **Actual Results** | | **Test Results** |
| Tasks are executed according its priority no matter whether or not they have the same priority | | PASS |
| **Comments** | | |
| Task priorities are configured according requirement 2.14 initially, but tasks sharing the same priority according requirement 2.15, are fully supported. See Figure 15 | | |

Os\_TaskCfg.c

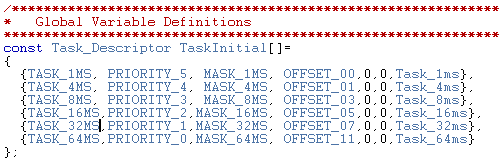
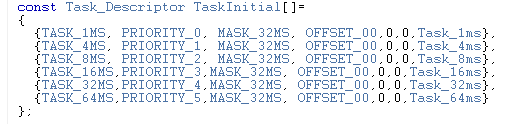


Figure 15

# Integration Test Cases

|  |  |  |
| --- | --- | --- |
| **Test Case** | **ID** | **Status** |
| Running project | **2.9** | **Done** |
| **Requirements covered** | | |
| 2.16, 2.17 | | |
| **Test Procedure** | | |
| Set a pin level to high when tasks start its execution. Set a pin level low when tasks end its execution. | | |
| **Expected Results** | | |
| Pin out level must be set high every time tasks start its execution, and must be set low every time the tasks ends its execution. | | |
| **Actual Results** | | **Test Results** |
| See the next figures which contains different configurations for the task properties. | | PASS |
| **Comments** | | |
| Each result is composed with two images, the first shows the configuration of the task and the second shows the result and order in which the task were executed. | | |

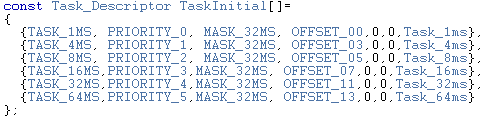
Next figures show each result.



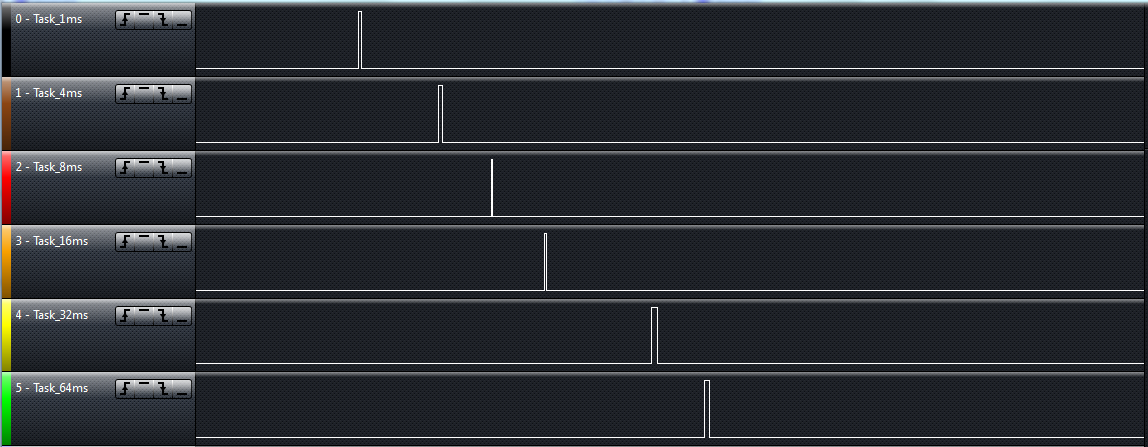
Code Test 1



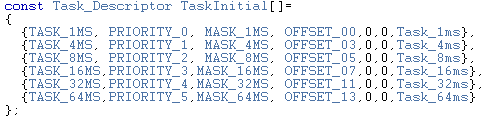
Result Test 1



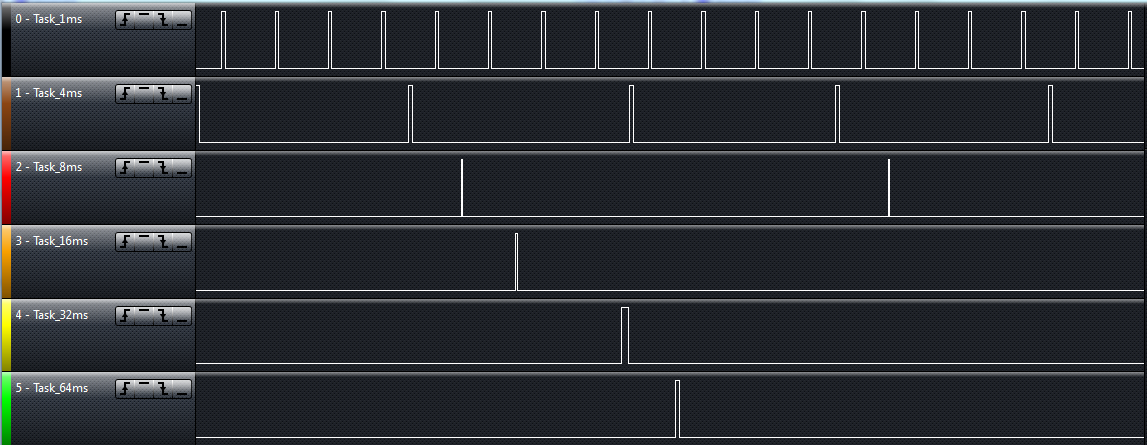
Code Test 2



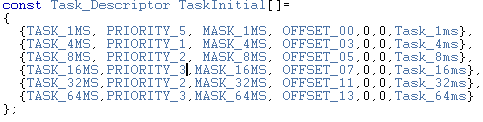
Result Test 2



Code Test 3



Result Test 3



Code Test 4



Result Test 4

|  |  |  |
| --- | --- | --- |
| **Test Case** | **ID** | **Status** |
| Running project | **2.10** | **Done** |
| **Requirements covered** | | |
| 2.18 | | |
| **Test Procedure** | | |
| Define the basic states Suspended, Ready & Running for all Tasks States. | | |
| **Expected Results** | | |
| Tasks states will be transitioning the basic states Suspended, Ready & Running. | | |
| **Actual Results** | | **Test Results** |
| See Figure 16 & Figure 17. | | PASS |
| **Comments** | | |
| Tasks will be set in suspended state every time they’re called to terminate in TerminateTask API.  Tasks in ready state will be set every time they’re put into the priority buffer by ActivateTask API.  Tasks in running state are set every time they’re called inside the Dispatcher API. | | |

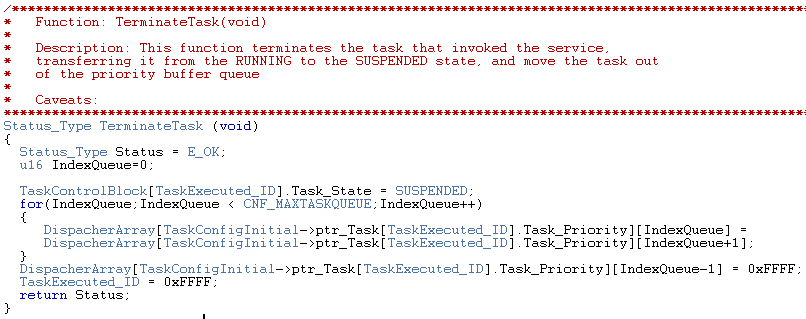
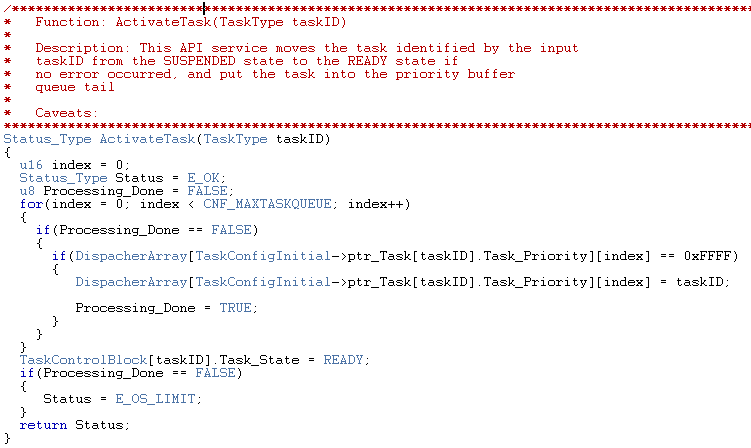


Figure 16



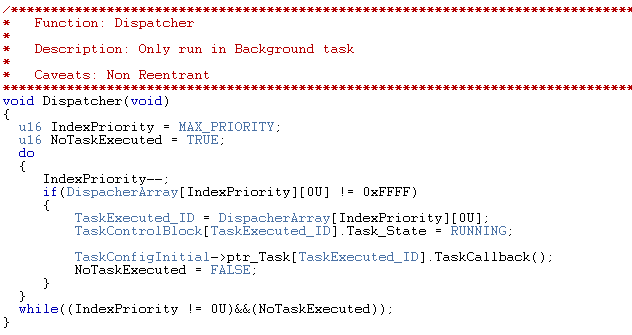


Figure 17

|  |  |  |
| --- | --- | --- |
| **Test Case** | **ID** | **Status** |
| Running project | **2.11** | **Done** |
| **Requirements covered** | | |
| 2.19 | | |
| **Test Procedure** | | |
| Place ready tasks IDs into its correspondent priority buffer. | | |
| **Expected Results** | | |
| Tasks IDs are placed into its correspondent priority buffer every time they turn ready. | | |
| **Actual Results** | | **Test Results** |
| Tasks IDs are placed into its correspondent priority buffer every time they turn ready. As figure 18 and 19. | | PASS |
| **Comments** | | |
|  | | |

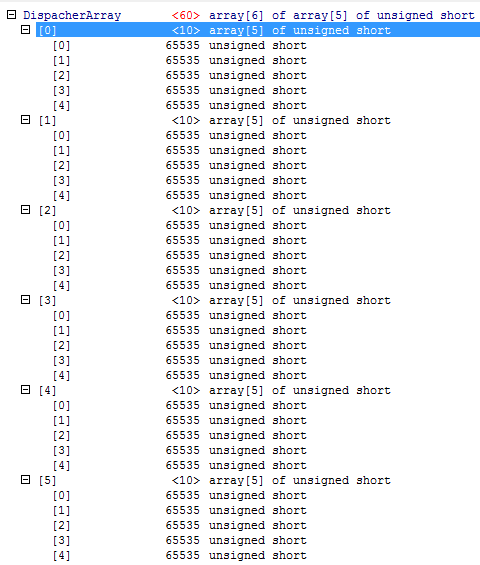


Figure 18

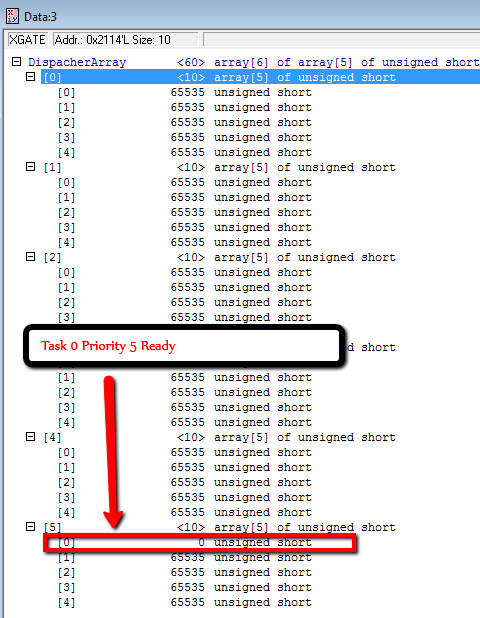


Figure 19

|  |  |  |
| --- | --- | --- |
| **Test Case** | **ID** | **Status** |
| Running project | **2.12** | **Done** |
| **Requirements covered** | | |
| 2.20 | | |
| **Test Procedure** | | |
| Move out tasks which transition from ready to suspended from its correspondent priority buffer | | |
| **Expected Results** | | |
| Tasks to be suspended are moved out from its last priority buffer. | | |
| **Actual Results** | | **Test Results** |
| Tasks to be suspended are moved out from its last priority buffer. As figure 20 and 21 | | PASS |
| **Comments** | | |
|  | | |

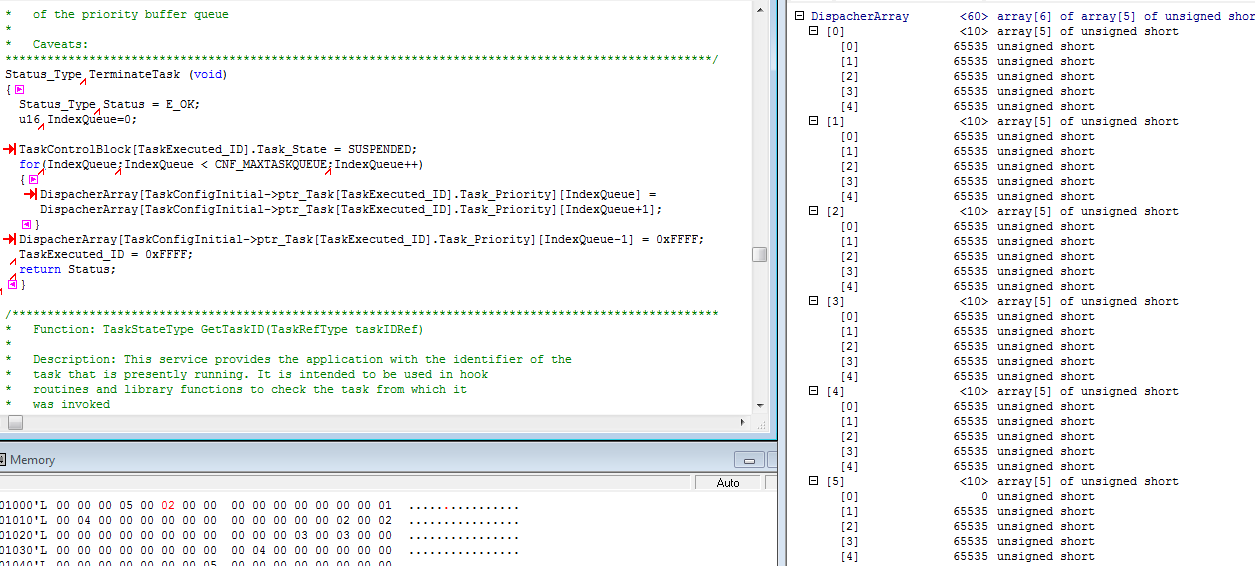


Figure 20

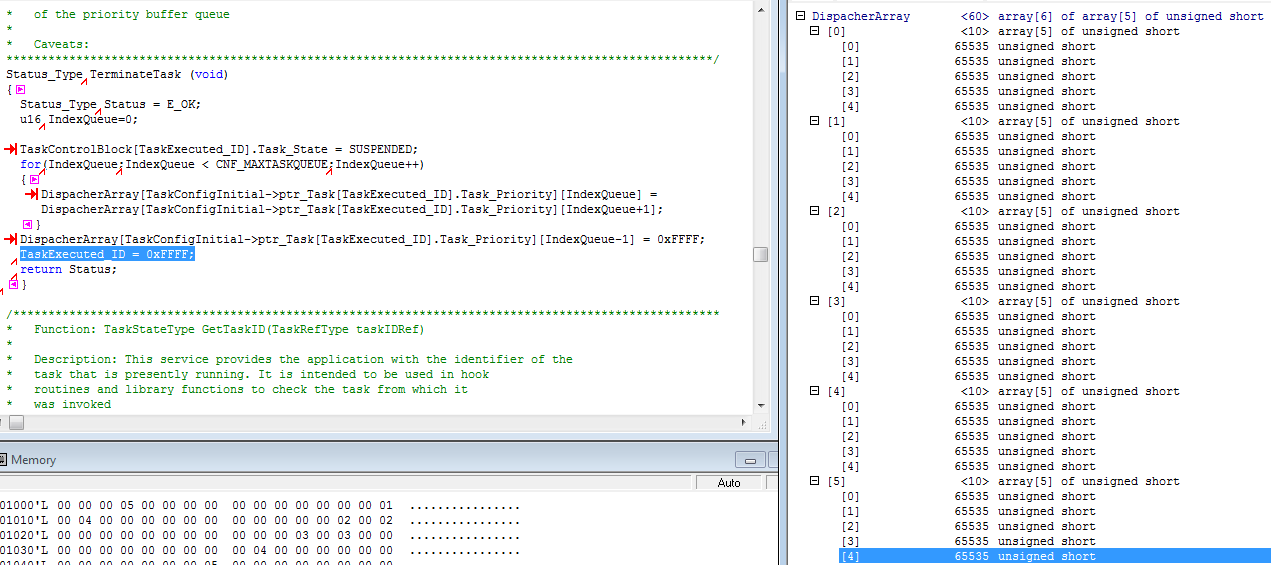


Figure 21

|  |  |  |
| --- | --- | --- |
| **Test Case** | **ID** | **Status** |
| Internal requirements | **2.13** | **Done** |
| **Requirements covered** | | |
| 2.21 | | |
| **Test Procedure** | | |
| Include memory allocation driver from previous projects. | | |
| **Expected Results** | | |
| Memory allocation will be available for usage in Task manager project. | | |
| **Actual Results** | | **Test Results** |
| Memory allocation will be available for usage in Task manager project. As figure 22 and 23 | | PASS |
| **Comments** | | |
|  | | |

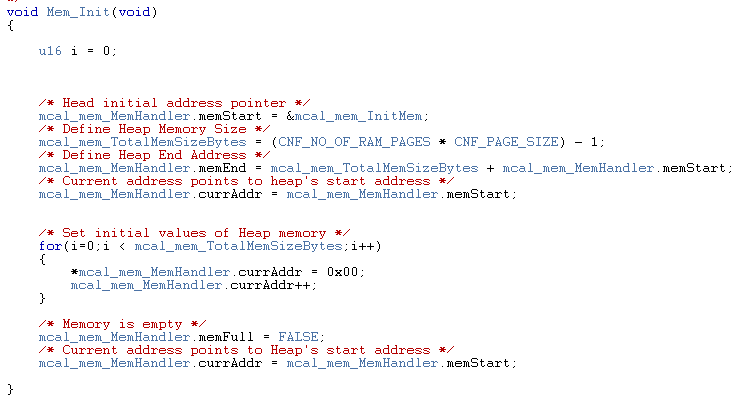


Figure 22



Figure 23

|  |  |  |
| --- | --- | --- |
| **Test Case** | **ID** | **Status** |
| Internal requirements | **2.14** | **Done** |
| **Requirements covered** | | |
| 2.22 | | |
| **Test Procedure** | | |
| Reserve memory for control block structure using memory allocation. | | |
| **Expected Results** | | |
| Reserved memory section is initialized to 0 and its size is the control block structure size. | | |
| **Actual Results** | | **Test Results** |
| Reserved memory section is initialized to 0 and its size is the control block structure size. As figure 24 and 25 | | PASS |
| **Comments** | | |
|  | | |

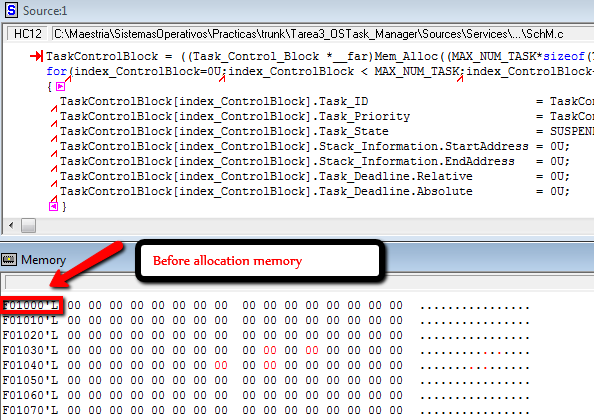


Figure 24

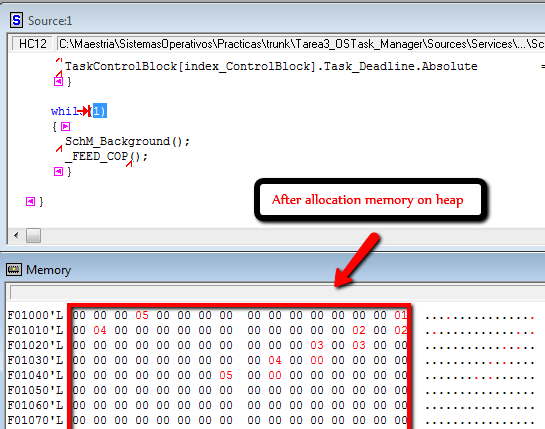


Figure 25