### **Lab 1 Assignment Questions**

1. **(1 point) Which global variable has the os\_active\_TCB array length information?**

The global variable *os\_maxtaskrun* contains the *os\_active\_TCB* array length information.

1. **(1 point) What does os\_active\_TCB array in RTX\_lib.c contain?**

The *os\_active\_TCB* array contains references to all the active task pointers.

1. **(1 point) Which global variable has the *os idle task* TCB information?**

The global variable *os\_idle\_TCB* contains the os idle task TCB information.

1. **(1 point) Is the TCB of the *os idle task* an element in the os\_active\_TCB?**

No, because *os\_active\_TCB* only stores active tasks; idle tasks are considered inactive.

1. **(1 point) Read rt\_get\_TID() code in rt\_Task.c file. Assume that a non-idle task has a task ID of n, what is the index of this task’s TCB in the os\_active\_TCB array?**

The index of a task with task ID n will be n-1 in *os\_active\_TCB* array.

1. **Read the rt\_TypeDef.h file and answer the following questions.** 
   * **(1 point) What is the purpose of the p\_lnk variable in the struct OS\_TCB?**

The *p\_lnk* variable is the pointer to the linked list for the ready/sem. waitlist of tasks.

* + **(1 point) What is the purpose of tsk\_stack and stack variables in struct OS\_TCB?**

*tsk\_stack* is a pointer to the top of the task stack; *stack* is a pointer to beginning of the task stack memory block.

* + **(1 point) If you have a variable with type of struct OS\_XCB \*, will casting this variable to struct OS\_TCB \* keep the p\_lnk field?**

Casting should result in the variable retaining the *p\_lnk* field, because both structs have that *p\_lnk* field.

1. **Read the rt\_Task.c and RTX\_lib.c files and answer the following question.**
   * **(2 points) What is the purpose of variables mp\_tcb and mp\_stk? You may want to search the entire source code to further explore how these variables are set and used (see Figure 1.28).**

*mp\_tcb* is a memory pool for TCB allocation; *mp\_stk* is a memory pool for system stack allocation.

1. **Read Section C.2.3 and study rt\_init\_stack() and rt\_get\_PSP() functions source code in HAL\_CM3.c file and answer the following questions.**
   * **(2 points) Which registers are saved on the task stack?**

The registers *R0* - *R15* are saved onto the task stack.

* + **(2 points) Which bits of which global variable contain the default task stack size in bytes?**

The 8 least significant bits of the *os\_stackinfo* variable contain the default task stack size.

* + **(2 points) How to determine the memory address of the first item that is pushed onto a task stack?**

The memory address of the first item that is pushed onto a task can be accessed by referencing *p\_TCB*🡪*stack*[*size* - 1].

* + **(2 points) For a task that is *not* in RUNNING state, how to determine the memory address of the last item that is pushed onto its task stack (i.e. the top of the stack)?**

Using *p\_TCB*->*tsk\_stack*.

* + (2 points) For a task that is in RUNNING state, how to determine the memory address of the last item that is pushed onto its task stack (i.e. the top of the stack)?

By calling the function *rt\_get\_PSP*().