1.

Task ID is be defined using the OS\_TID type, which is an unsigned integer. The index of the task's TCB is this integer value -1.

The index of this task is n-1.

2.

p\_lnk is the link pointer for ready/sem. wait list.

ret\_val is the return value upon completion of a wait .

tsk\_stack is Current task Stack pointer (R13).

Stack is the pointer to Task Stack memory block.

Yes, casting will keep the p\_lnk field, because the p\_lnk field is in both structures and they point to the same thing. Therefore casting a XCB to a TCB has no affect on p\_lnk.

3.

mp\_tcb is a data structure of memory pool tab location.

mp\_stk is the same thing only for memory pool stack allocation

4.

On the task stack, there are the General Purpose Registers R0 - R12 and the LR register.

To determine the start address: call p\_tcb -> stack[0]

To determine the end address: call p\_tcb ->stack[size]

where size is the total size of the stack

To return the current stack pointer of a task that is not in Running state:

call p\_tcb -> tsk\_stack

To return the current stack pointer of a task that is in Running state:

call rt\_get\_PSP()

5.

To declare a memory pool:

\_declare\_box(mypool, 12, 20);

To initialize a memory pool:

\_initialize\_box(mypool, sizeof(mypool), 12);

The corresponding kernel function of \_alloc\_box() is:

rt\_alloc\_box(void \*box\_mem);

The corresponding kernel function of \_free\_box() is:

rt\_free\_box (void \*box\_mem);

6.

rt\_put\_prio() puts a p\_task into a ordered list based on its priority.

rt\_get\_first() gets the task at the head of the ordered list (with the highest priority)

7.

rt\_block(): block running task and choose next ready task.

rt\_dispatch(): dispatch next task if any identified or dispatch highest ready task.

8.

The rt\_mbx\_wait() does not have a line to set the return value to OS\_R\_MBX because doing so would block the os\_mbx\_wait function itself.

9. The two queues use linked list.