Read the RTX Library Reference in the Keil IDE Help (see 1.12) and answer the

following questions:

\_ In an RTX application, how to declare a memory pool named "mympool" that

has 20 blocks of memory with block size of 12 bytes?

Use the \_declare\_box macro or \_declare\_box8 for 8 byte alignment

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

Defined in RTX\_Config.h by:

#define \_declare\_box(pool,size,cnt) U32 pool[(((size)+3)/4)\*(cnt) + 3]

\_ Write one line of code to initialize the "mympool".

\_init\_box (mympool, 12, 20);

What the fuck is a kernel function?! Refers to functions prefixed with rt\_

\_ What is the corresponding kernel function of \_alloc\_box()?

void \*rt\_alloc\_box (void \*box\_mem) in rt\_MemBox.c

\_ What is the corresponding kernel function of \_free\_box()?

int rt\_free\_box (void \*box\_mem, void \*box)

6. Read the rt\_TypeDef.h \_le. What is the purpose of ret\_val in the struct OS\_TCB?

U32 ret\_val; /\* Return value upon completion of a wait \*/

7. Read the rt\_List.c \_le and answer the following questions:

\_ What does the rt\_put\_prio() function do?

void rt\_put\_prio (P\_XCB p\_CB, P\_TCB p\_task) {

/\* Put task identified with "p\_task" into list ordered by priority. \*/

/\* "p\_CB" points to head of list; list has always an element at end with \*/

/\* a priority less than "p\_task->prio".

Inserts P\_TCB argument into a priority queue

\_ What does the rt\_get\_first() function do?

P\_TCB rt\_get\_first (P\_XCB p\_CB) {

/\* Get task at head of list: it is the task with highest priority. \*/

/\* "p\_CB" points to head of list. \*/

You may want to further explore other functions in the rt\_List.c \_le.

Read the rt\_Task.c \_le and answer the following questions:

\_ What does the rt\_block() function do?

Blocks the running task and chooses the next task from the priority queue of ready tasks (os\_rdy). Sets some timeout time, and assigns the (previously running) task a new state.

void rt\_block (U16 timeout, U8 block\_state) {

/\* Block running task and choose next ready task. \*/

/\* "timeout" sets a time-out value or is 0xffff (=no time-out). \*/

/\* "block\_state" defines the appropriate task state \*/

P\_TCB next\_TCB;

if (timeout) {

if (timeout < 0xffff) {

rt\_put\_dly (os\_tsk.run, timeout);

}

os\_tsk.run->state = block\_state;

next\_TCB = rt\_get\_first (&os\_rdy);

rt\_switch\_req (next\_TCB);

}

}

\_ What does the rt\_dispatch() function do?

Takes in the input P\_TCB which is the task to be dispatched. If its priority is higher than the current running task, then the current running task is pre-empted (put to a READY state), and the new task is now running.

void rt\_dispatch (P\_TCB next\_TCB) {

/\* Dispatch next task if any identified or dispatch highest ready task \*/

/\* "next\_TCB" identifies a task to run or has value NULL (=no next task) \*/

if (next\_TCB == NULL) {

/\* Running task was blocked: continue with highest ready task \*/

next\_TCB = rt\_get\_first (&os\_rdy);

rt\_switch\_req (next\_TCB);

}

else {

/\* Check which task continues \*/

if (next\_TCB->prio > os\_tsk.run->prio) {

/\* preempt running task \*/

rt\_put\_rdy\_first (os\_tsk.run);

os\_tsk.run->state = READY;

rt\_switch\_req (next\_TCB);

}

else {

/\* put next task into ready list, no task switch takes place \*/

next\_TCB->state = READY;

rt\_put\_prio (&os\_rdy, next\_TCB);

}

}

}