2.4.1 Questions

Please answer the following questions before you proceed with coding.

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

following questions:

\_ (1 point) In an RTX application, how to declare a memory pool named "mympool"

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

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

\_ (1 point) Write one line of code to initialize the "mympool".

**\_init\_box (mympool, sizeof(mympool), 12);**

\_ (1 point) What is the corresponding kernel function of \_alloc\_box()?

**/\*--------------------------- rt\_alloc\_box ----------------------------------\*/**

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

**/\* Allocate a memory block and return start address. \*/**

**void \*\*free;**

**#ifndef \_\_USE\_EXCLUSIVE\_ACCESS**

**int irq\_dis;**

**irq\_dis = \_\_disable\_irq ();**

**free = ((P\_BM) box\_mem)->free;**

**if (free) {**

**((P\_BM) box\_mem)->free = \*free;**

**}**

**if (!irq\_dis) \_\_enable\_irq ();**

**#else**

**do {**

**if ((free = (void \*\*)\_\_ldrex(&((P\_BM) box\_mem)->free)) == 0) {**

**\_\_clrex();**

**break;**

**}**

**} while (\_\_strex((U32)\*free, &((P\_BM) box\_mem)->free));**

**#endif**

**return (free);**

**}**

\_ (1 point) What is the corresponding kernel function of \_free\_box()?

**/\*--------------------------- rt\_free\_box -----------------------------------\*/**

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

**/\* Free a memory block, returns 0 if OK, 1 if box does not belong to box\_mem \*/**

**#ifndef \_\_USE\_EXCLUSIVE\_ACCESS**

**int irq\_dis;**

**#endif**

**if (box < box\_mem || box >= ((P\_BM) box\_mem)->end) {**

**return (1);**

**}**

**#ifndef \_\_USE\_EXCLUSIVE\_ACCESS**

**irq\_dis = \_\_disable\_irq ();**

**\*((void \*\*)box) = ((P\_BM) box\_mem)->free;**

**((P\_BM) box\_mem)->free = box;**

**if (!irq\_dis) \_\_enable\_irq ();**

**#else**

**do {**

**\*((void \*\*)box) = (void \*)\_\_ldrex(&((P\_BM) box\_mem)->free);**

**} while (\_\_strex ((U32)box, &((P\_BM) box\_mem)->free));**

**#endif**

**return (0);**

**}**

2. (2 points) Read the rt\_TypeDef.h \_le. What is the purpose of ret\_val in the

struct OS\_TCB?

**This field is used to return value upon completion of a wait**

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

\_ (2 points) What does the rt\_put\_prio() function do?

**This function use two arguments, p\_CB is the pointer to the list you want to put the task in, and p\_task is the pointer to the task that you want to move to the list. Also this function will sort the tasks by their priority.**

\_ (2 points) What does the rt\_get\_first() function do?

**This function has one argument which is the pointer to the head of the list. This function will return the task on the top, which has the highest priority.**

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

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

\_ (2 points) What does the rt\_block() function do?

**Block running task and choose next ready task. There are two arguments, timeout is for setting the value of time out 0xffff is no time out), block\_state defines appropriate state.**

\_ (2 points) What does the rt\_dispatch() function do?

**This function is used to send the next task which is ready for running to the processor. If this function has higher priority, then run it first. Otherwise, put it into ready list.**

5. (4 points) How to set the return value of a function becomes a bit tricky when context

switching is involved. One such example is os\_mbx\_wait() function. This function

will return OS\_R\_MBX if the task has waited until a message was put in the mailbox

(i.e. the task was blocked to wait for a message to arrive and then unblocked when

the message arrives). Read the rt\_Mailbox.c \_le and \_nd the lines where the return

value of OS\_R\_MBX is set. Why the corresponding kernel function rt\_mbx\_wait()

does not have a line to set the return value to OS\_R\_MBX? You may skip the code in

functions isr\_mbx\_receive() and rt\_mbx\_psh() for the purpose of completing this

assignment.

**OS\_R\_MBX will only be set when the tasks receiving is blocked and resumed (the task was blocked to wait for a message to arrive and then unblocked when the message arrives). Context switching is happening in the HAL\_CM3.c in the “config” file rather than in c code in the Kernel level. Only the sender can set the value of OS\_R\_MBX and that will only happen when the tasks receiving is blocked and resumed. Since the context switching happening in the HAL\_CM3.c in the “config” file, we are out of the kernel and cannot set** **the return value to OS\_R\_MBX.**

6. (2 points) To block a task, you will need to create a queue that the blocked tasks

can stay. There are two global queues for now in the kernel and they are os\_rdy and

os\_dly. What data structure do these two queues use?

**It is a linked list.**