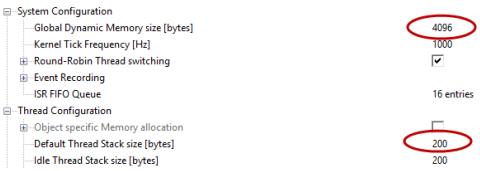
Exercise 3 Memory Model

In this exercise we will create a thread with a custom memory allocation and also create a thread with a static memory allocation.

In the Pack Installer select “Ex memory model” and copy it to your tutorial directory.

This exercise uses the same two LED flasher threads as the previous exercise.

Open cmsis::rtx_config.c



The threads are allocated memory from the global dynamic memory pool and by default each thread is allocated 200 bytes

When we create led-thread1 we pass the attribute structure which has been modified to create the thread with a custom stack size of 1025 bytes

```
static const osThreadAttr_t ThreadAttr_LED1 = {
        "LED_Thread_1",
        NULL,      //attributes
        NULL,      //cb memory
        NULL,      //cb size
        NULL,      //stack memory
        1024,      //stack size          This memory is allocated from the global memory pool
        osPriorityNormal,
        NULL,      //trust zone id
        NULL       //reserved
};
```

The second thread is created with a statically defined thread control block and a statically defined stack space. First we need to define an array of memory for the stack space;

```
static uint64_t LED2_thread_stk[64];
```

Followed by a custom RTX thread control block;

```
static osRtxThread_t LED2_thread_tcb;
```

The custom type osRtxThread is defined in rtx_os.h

Now we can create a thread attribute which statically allocates the both the stack and the task control block;

```
static const osThreadAttr_t ThreadAttr_LED2 = {
        "LED_Thread_2",
        NULL,                       //attributes
        &LED2_thread_tcb,           //cb memory
        sizeof(LED2_thread_tcb),    //cb size
        &LED2_thread_stk[0],         //stack memory     Here the control block and user stack space are statically allocated
        sizeof(LED2_thread_stk),    //stack size
        osPriorityNormal,
        NULL,               //trust zone id
        NULL                //reserved
};
```

Build the code.

Start the debugger and check it runs

The statically allocated thread will not appear in the RTOS component viewer as the custom memory allocation is not detected

Exit the debugger

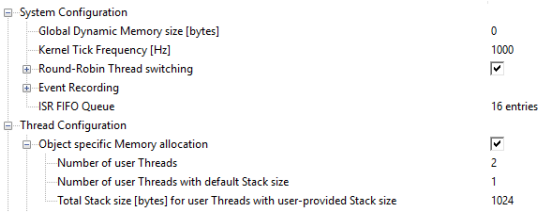
In the CMSIS:RTX_Conf.c file we can change the memory model to use “Object Specific” memory allocation.

Set the Global Dynamic memory size to zero

In thread configuration enable the Object specific memory model

Set the number of threads to two

Number of user threads wit default stack size to 1 and total stack size for threads with use provided stack to 1024.



In total we have three user threads but one has statically allocated memory so our thread object pool only needs to accommodate two. One of those threads (Led_thread1) has a custom stack size

of 1024 bytes. We need to provide this information to the RTOS so it can work out the total amount of memory to allocate for thread use.

Enable the MUTEX object

Set the number of mutex objects to 5

We will use mutexes later but they are concerned with protecting access to resources. The RTOS creates a number to protect access to the run time ‘C’ library from different threads.

Build the code

Start the debugger

Run the code

Now we have one thread using statically located memory and object using object specific memory.