Lab Sheet 4: Real-Time OS

See deadlines on QMPlus

1 Aims

The aim of this lab is to make use of the ARM real-time operating system (RTOS):

- Understand and be able to use the 'thread' abstraction offered by the RTOS
- Be able to use RTOS 'event' functions.
- Be able to use RTOS time functions.

The Real-Time OS can be described as CMSIS-RTOS or RTX.

- CMSIS-RTOS is an API for an RTOS
- RTX is an implementation of this API.

The API documentation is at:
http://www.keil.com/pack/doc/CMSIS/RTOS/html/index.html

A new version of RTX – RTX5 – has recently been introduced and with it a revised API, CMSIS-RTOS2. This lab uses CMSIS-RTOS version 1, provided by a compatibility layer.

1.1 Overview of Activities

Perform the following activities and answer the questions given in section 5.

1. Activity 1: Download and review the sample code.
2. Activity 2: Modify the code for some new requirements, perhaps using one of the designs described in the lecture.
3. Activity 3: Implement code for a second variant, investigating suitable functions in the CMSIS-RTOS API.

You must also answer the questions (see Section 5) on the QMPlus quiz and demonstrate your program in the scheduled lab time. *You are recommended to look at the questions as you go along: some are about the given code and answering them will help you complete the rest of the lab.*

2 Activity 1: Download, Read, Compile and Run the Sample Project

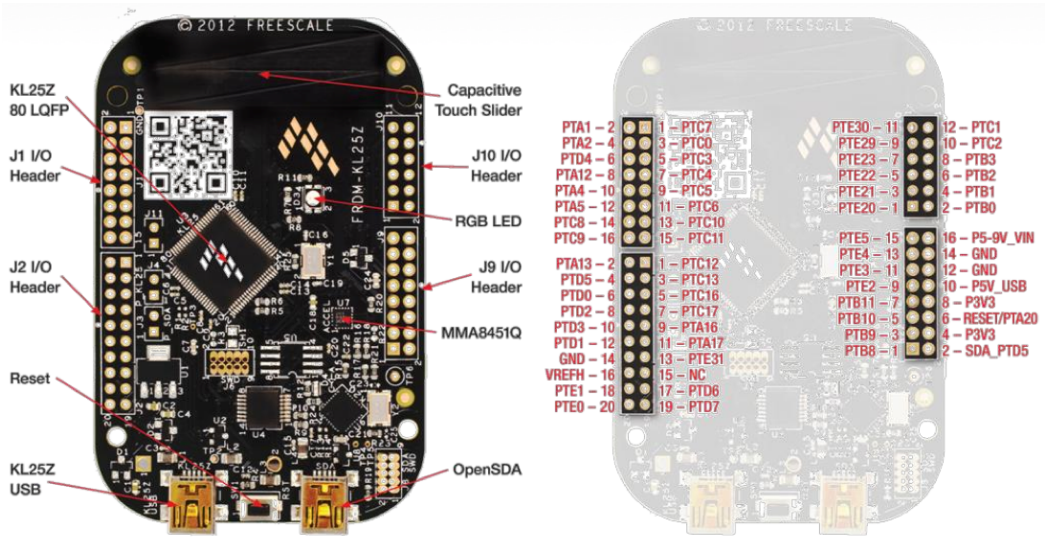
Download the sample project. This project uses two threads, t_button and t_led, to flash the red LED and turn the green LED on/off when the button is pressed. The two threads are independent: the flashing red LED is not affected by the button.

Pause and read the complete code carefully before going on. *Answer the questions on the question sheet about the code.*

2.1 Button Connection

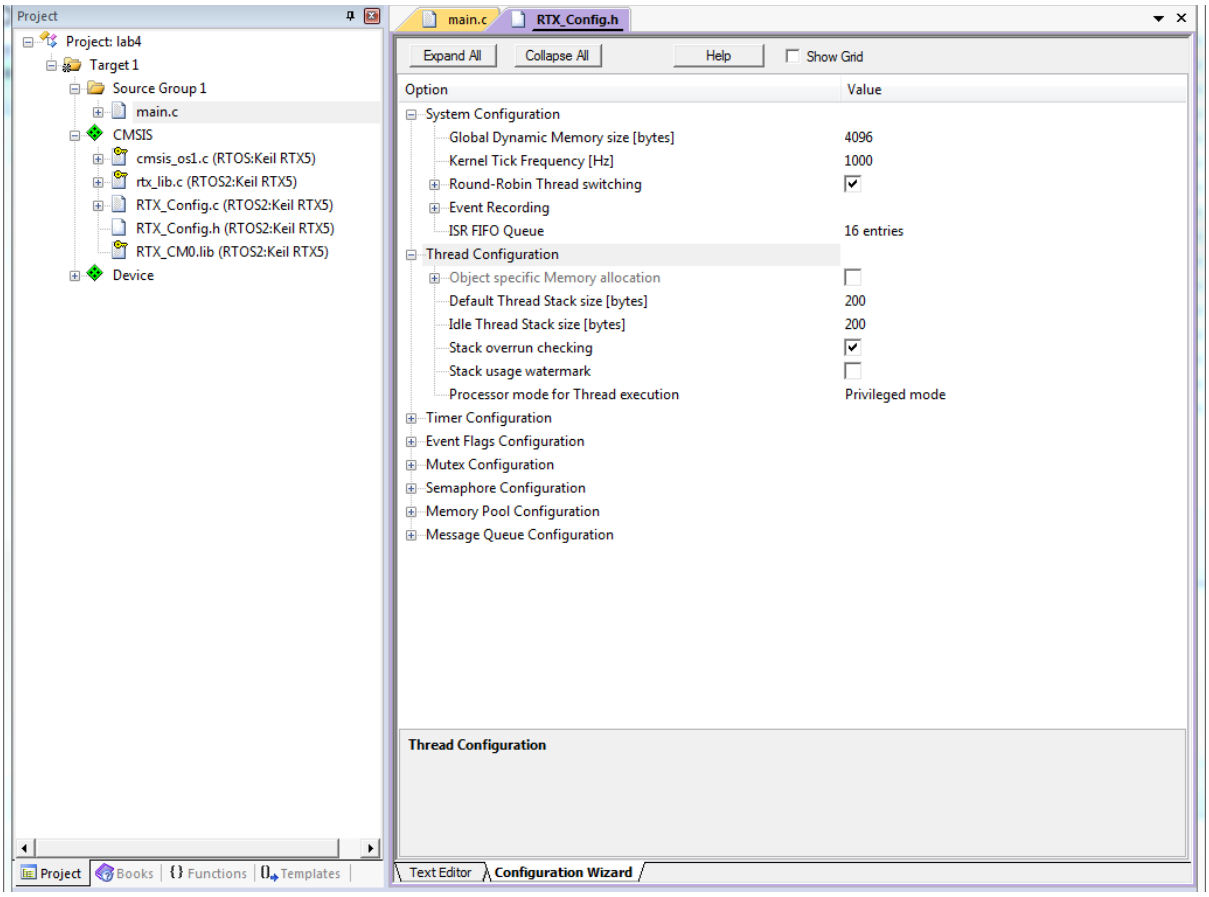
The button is connected in the same way as in Lab 2. The pins are:

- GPIO input to port D bit 6, which is J2 pin 17
- Ground (0v), which is J2 pin 14 (for example)



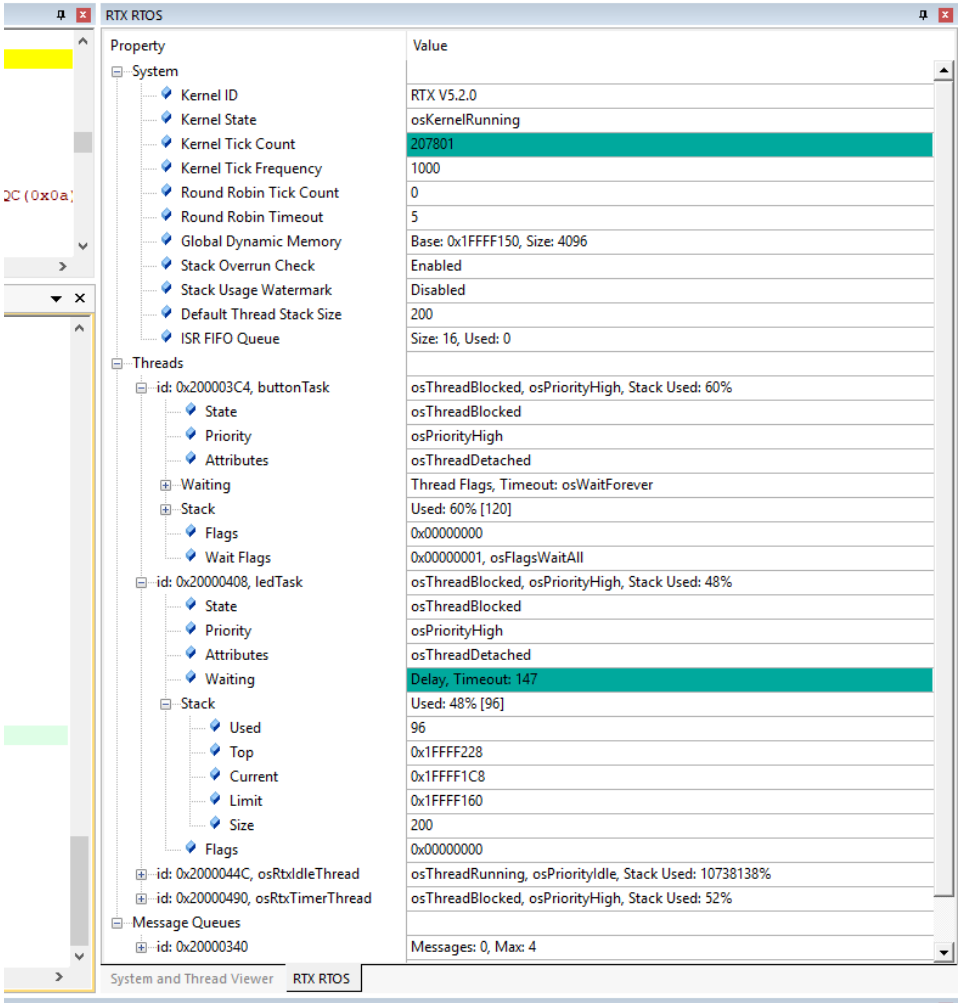
2.2 RTOS Configuration

A configuration wizard allows the RTX_Config.h file to be viewed and edited. Be careful not to change anything!



2.3 Viewing the RTOS Threads in the Debugger

The IDE can show running RTOS information in the debugger: use View/Watch/RTX. The application must be running. (There is also a System and Thread Viewer, but it does not work with RTX5 in the current release of µVision.)



3 Activity 2: Start / Stop Flashing

Make a copy of the project and modify it so that it meets the following requirements. The modified project must still use the RTOS, rather than a cyclic system design.

1. The LEDs flash red then green then blue in a cycle, with one colour always showing, each for 3 seconds.
2. Pressing the button causes the flashing of the LEDs to stop / restart.
3. There should be no noticeable delay to the button being pressed.
4. When the flashing restarts, the LED that was alight when flashing stops should light up again.

It is recommended that you make a **plan** for completing this change. Possible steps are:

1. Clarify the requirements. Sketch a state transition model for the behaviour.
2. Design the solution. Decide on the number of threads and the events and delays used in each thread (and the `osXX` functions needed)[1].
3. Look up the `osXX` functions needed and insert fragments of code.
4. Create any new threads, each with: a) a variable b) function for thread behaviour c) thread initialisation.

Viva Required [8 marks]

A demonstrator will witness the viva. Ensure that both you and the demonstrator sign the viva sheet.

The code should be uploaded – see instructions on QMPlus.

4 Activity 3: Set Flash Rate

Make a copy of the project and modify it so that it meets the following requirements. The modified project must still use the RTOS, rather than a cyclic system design.

1. When started, no LEDs flash.
2. When the button is pressed, an LED lights and stays alight.
3. On the second button press, the flashing starts. The system records the time between the two presses and uses this as the period of the flashing.
4. Further button presses are ignored.

It is likely that this will require new API call: see http://arm-software.github.io/CMSIS_5/RTOS/html/functionOverview.html and see if it is possible to access SysTick.

Viva Required [8 marks]

A demonstrator will witness the viva. Ensure that both you and the demonstrator sign the viva sheet.

The code should be uploaded – see instructions on QMPlus.

[1] Two possible designs were described in the lecture in week 4.

5 Question Sheet

Answers to these questions should be completed on the QMplus quiz.

5.1 Questions About the Given Code and the Lab Activities [2 marks each]

Relates to	Questions
Section 2 Activity 1	In the provided code, briefly describe what the two tasks 't_led' and 't_button' are used for.
Section 2 Activity 1	When the provided code runs, how many threads are active? Briefly describe what each is doing (if not mentioned above).
Section 2 Activity 1	Why does each thread have a stack (as shown in the debugger View/Watch/RTX)? What is the stack used for and where is it?
Section 3 Activity 2	In the code you have written for activity 2, explain what happens to the timing of the 3 sec period when the flashing stops/restarts.

5.2 Question about concepts

Answer the following questions concisely:

A simple micro-controller system flashes a pair of LEDs at different rates. Briefly contrast the way the system could implemented with and without an RTOS, using the following headings:

1. How the system is organised into several parts
2. How delays and timing are done
3. Whether the implementation is using concurrency.

[8 marks]