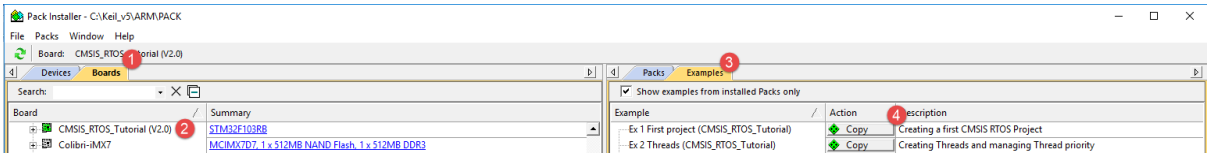
Exercise 1 A first CMSIS-RTOS2 project

This project will take you through the steps necessary to create and debug a CMSIS-RTOS2 based project.

First start the pack installer 

This can be done from within microvision from the main toolbar

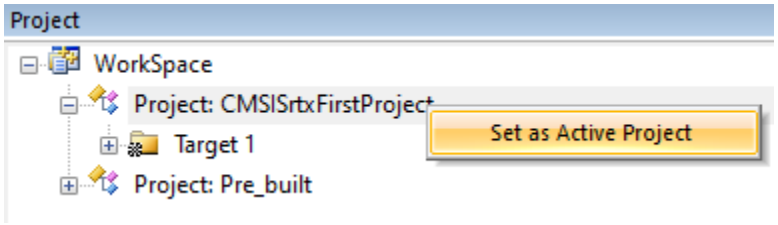


In the pack installer select the boards tab, then select the CMSIS-RTOS Tutorial

Next select the Examples tab and open the first example by pressing the copy button

This will open a project shell which has been setup for the STM32F103B. This is a basic Cortex-M3 microcontroller. In microvision there is a legacy simulator which has a full model for the STM32F103. This allows us to experiment with CMSIS-RTOS2 without the need for a specific hardware board.

This first project is a multi project workspace. The shell project is set as the active project. A pre built working project is included as a reference. If you want to build this project highlight the project, right click and select “Set as active project”. Any compile and debug actions will work on the active project.



Open the Run Time Environment (RTE) by selecting the green diamond on the toolbar 

The RTE allows you to configure the platform of software components you are going to use in a given project. As well as displaying the available components the RTE understands their dependencies on other components.

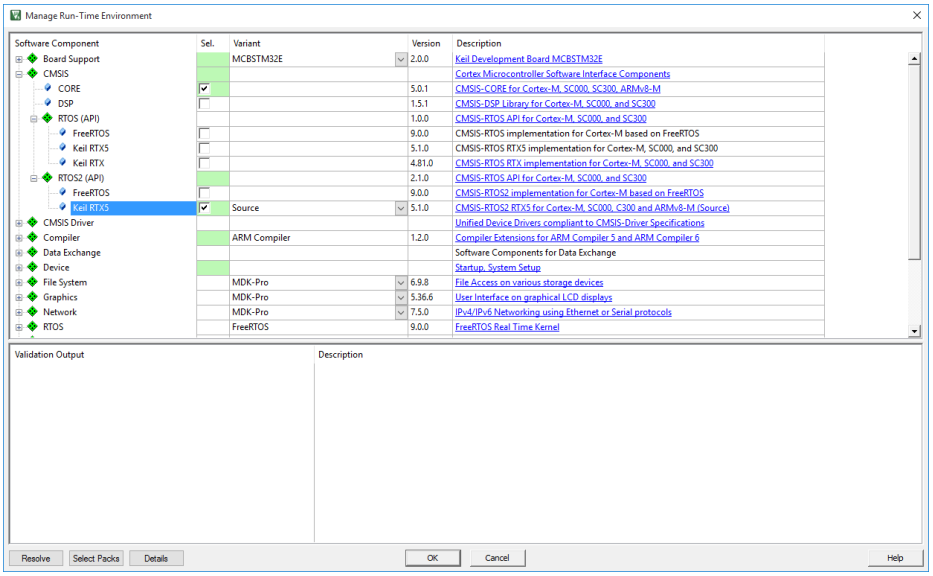


Fig 6 Add the RTOS

To configure the project for use with the CMSIS-RTOS2 Keil RTX, simply tick the CMSIS::RTOS2 (API):Keil RTX5 box.

Switch the Keil RTX5 dropdown variant box from 'Source' to 'Library'.

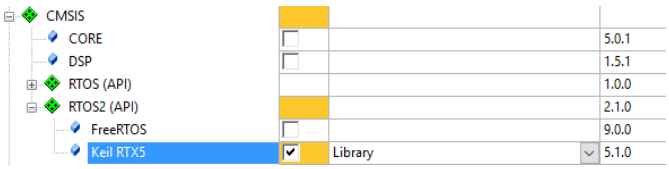


Fig 7 If the Sel column elements turn Orange then the RTOS requires other components to be added

This will cause the selection box to turn orange meaning that additional components are required. The required component will be displayed in the Validation Output window.

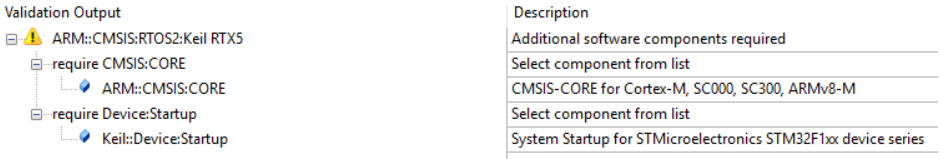


Fig 8 The validation box lists the missing components

To add the missing components you can press the Resolve button in the bottom left hand corner of the RTE.

This will add the device startup code and the CMSIS Core support. When all the necessary components are present the selection column will turn green.

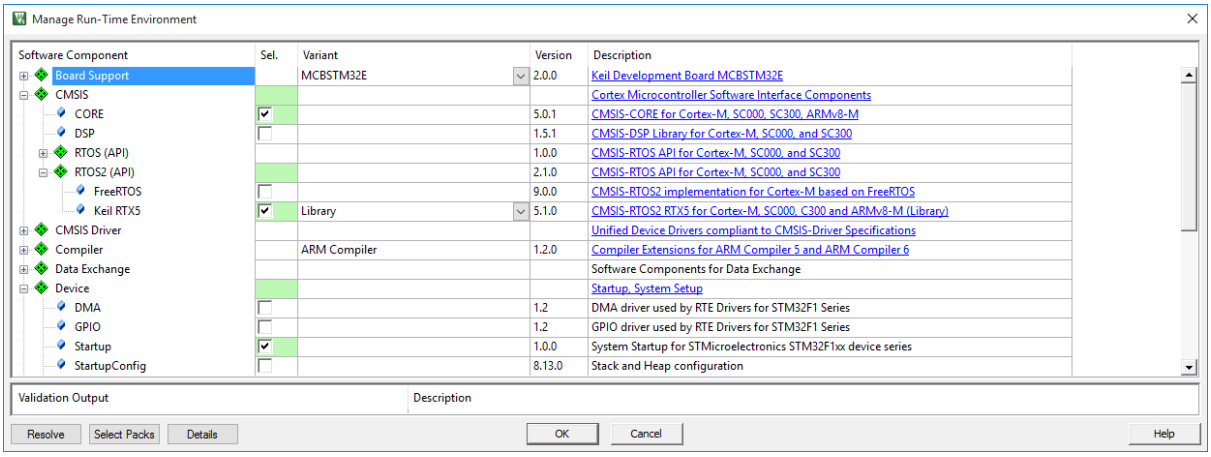


Fig 9 pressing the resolve button adds the missing components and the Sel. Column turns green

It is also possible to access a components help files by clicking on the blue hyperlink in the Description column.

The other RTOS options will be discussed towards the end of this tutorial.

Now press the OK button and all the selected components will be added to the new project

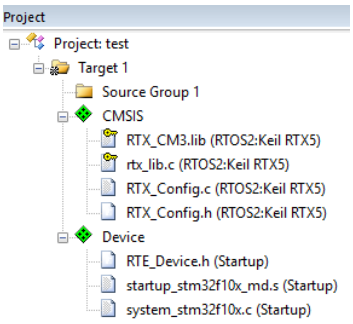


Fig 9 The configured project platform

The CMSIS components are added to folders displayed as a green diamond. There are two types of file here. The first type is a library file which is held within the tool chain and is not editable. This file is shown with a yellow key to show that it is 'locked' (read-only). The second type of file is a configuration file. These files are copied to your project directory and can be edited as necessary. Each of these files can be displayed as a text files but it is also possible to view the configuration options as a set of pick lists and drop down menus.

To see this open the RTX_Config.h file and at the bottom of the editor window select the 'Configuration Wizard' tab.

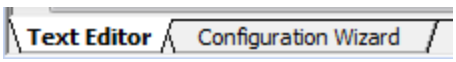


Fig 10 Selecting the configuration wizard

Click on Expand All to see all of the configuration options as a graphical pick list:

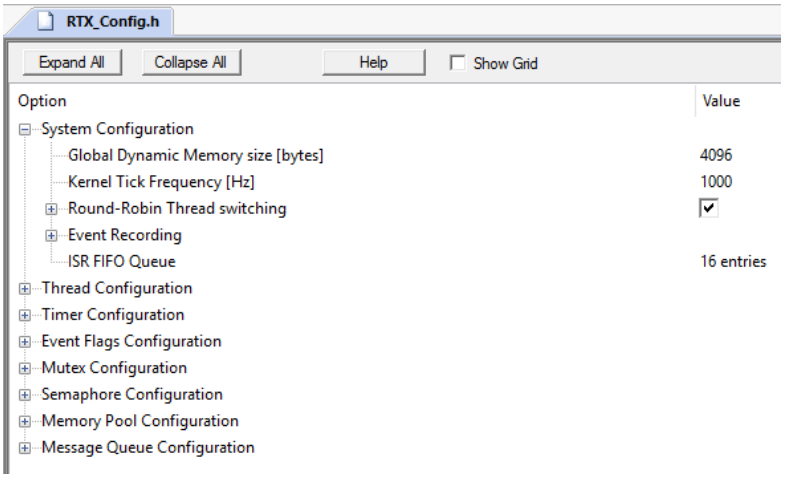


Fig 11 The RTX configuration options

For now it is not necessary to make any changes here and these options will be examined towards the end of this tutorial.

Our project contains four configuration files three of which are standard CMSIS files

File name	Description
Startup_STM32F10x_md.s	Assembler vector table
System_STM32F10x.c	C code to initialize key system peripherals, such as clock tree, PLL external memory interface.
RTE_Device.h	Configures the pin multiplex
RTX_Config.h	Configures Keil RTX

Table 2 Project configuration files

Now that we have the basic platform for our project in place we can add some user source code which will start the RTOS and create a running thread.

To do this right-click the 'Source Group 1' folder and select 'Add new item to Source Group 1'

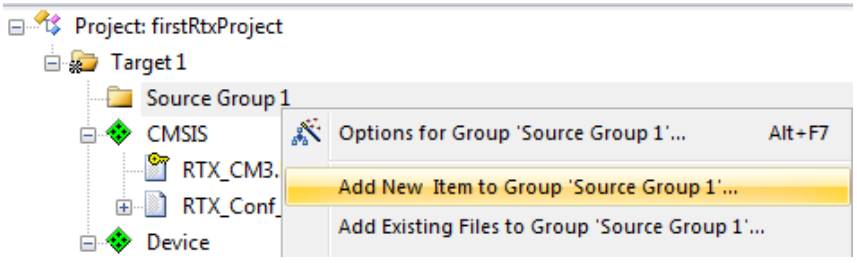


Fig 12 Adding a source module

In the Add new Item dialog select the 'User code template' Icon and in the CMSIS section select the 'CMSIS-RTOS 'main' function' and click Add

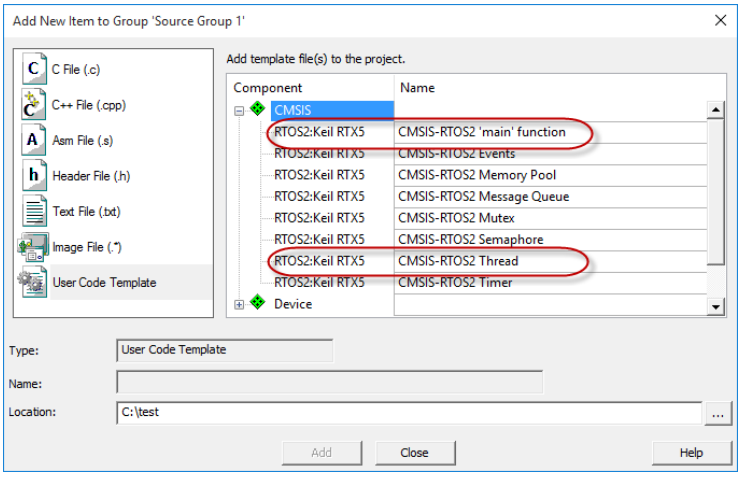


Fig 13 selecting a CMSIS RTOS template

Repeat this but this time select 'CMSIS-RTOS2 Thread'.

This will now add two source files to our project main.c and thread.c

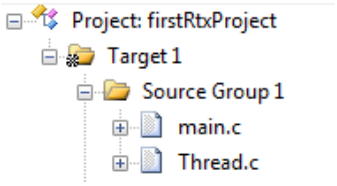


Fig 14 The project with main and Thread code

Open thread.c in the editor

We will look at the RTOS definitions in this project in the next section. For now this file contains two functions Init_Thread() which is used to start the thread running and the actual thread function.

Copy the Init_Thread function prototype and then open main.c

Main contains the functions to initialize and start the RTOS kernel. Then unlike a bare metal project main is allowed to terminate rather than enter an endless loop. However this is not really recommended and we will look at a more elegant way of terminating a thread later.

In main.c add the Init_Thread prototype as an external declaration and then call it after the osKernelInitialize() function as shown below.

```
extern int Init_Thread (void);

/*----------------------------------------------------------------------------

* Application main thread

*----------------------------------------------------------------------------*/

void app_main (void *argument) {

        Init_Thread ();

       for (;;) {}

}
```

Build the project (F7)

Start the debugger (Ctrl+F5)

This will run the code up to main

Open the Debug → View → Watch Windows →RTX RTOS

Start the code running (F5)

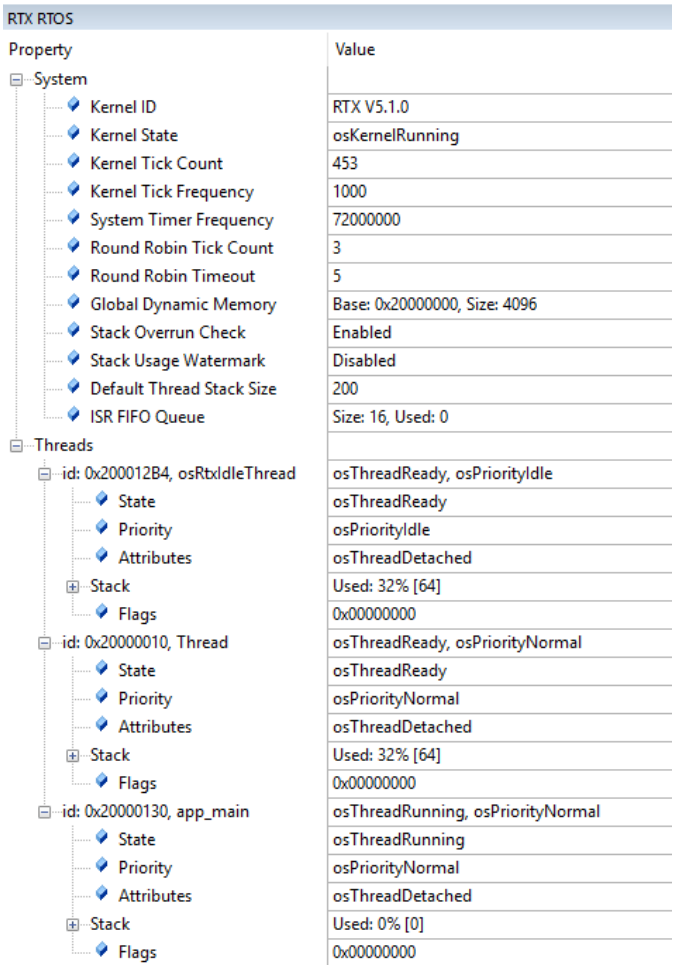


Fig 16 The RTX5 component viewer

This debug view shows all the running threads and their current state. At the moment we have three threads which are app_main, osRtxIdleThread and Thread.

This window is a component view which shows key variables in a software library (component). It is generated by an XML file. It is possible to create such a view for key variables in your application code. This is very useful if you have a long term project or code that you are going to give to a third party.

Exit the debugger

While this project does not actually do anything it demonstrates the few steps necessary to start using CMSIS-RTOS2.