

Exercise 8 Idle Thread

In the Pack Installer select “Ex 8 Idle” and copy it to your tutorial directory.

This is a copy of the virtual timer project.

Open the RTX_Config.c file and click the text editor tab

Locate the idle thread

```
__NO_RETURN void osRtxIdleThread (void *argument){  
  
    for (;;) {  
  
        //wfe();  
  
    }  
}
```

Build the code and start the debugger

Run the code and observe the activity of the threads in the event Viewer.

This is a simple program which spend most of its time in the idle demon so this code will be run almost continuously

Open the View → Analysis Windows → Performance Analyzer.

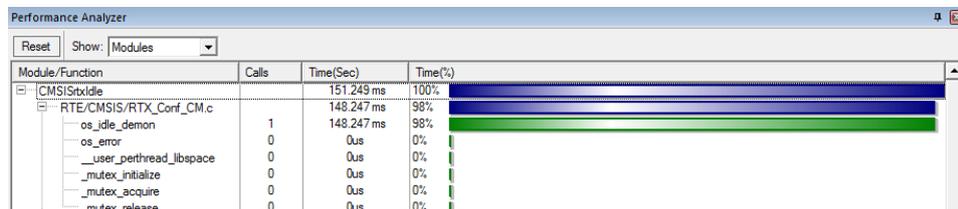


Fig 33 The performance analyser shows that most of the run time is being spent in the idle loop

This window shows the cumulative run time for each function in the project. In this simple project the idle thread is using most of the runtime because there is very little application code.

Exit the debugger

Remove the delay loop and the toggle instruction and add a `__wfe()` instruction in the for loop, so the code now looks like this.

```
__NO_RETURN void osRtxIdleThread (void *argument){  
  
    for (;;) {  
  
        __wfe();  
  
    }  
}
```

Rebuild the code, restart the debugger

Now when we enter the idle thread the `__wfe()` (wait for event) instruction will halt the CPU until there is a peripheral or SysTick interrupt.

Module/Function	Calls	Time(Sec)	Time(%)
CMSISrtxIdle		11.496 ms	2%
./rt_List.c		3.805 ms	1%
./rt_System.c		1.948 ms	0%
./rt_Robin.c		1.570 ms	0%
HAL_CM3.c		1.296 ms	0%
./rt_CMSIS.c		1.139 ms	0%
./rt_Task.c		1.091 ms	0%
RTE/CMSIS/RTX_Conf_CM.c		596.739 us	0%
os_idle_demon	1	596.739 us	0%
os_error	0	0us	0%
__user_perthread_libspace	0	0us	0%
__mutex_initialize	0	0us	0%
__mutex_acquire	0	0us	0%
__mutex_release	0	0us	0%

Fig 34 The `__wfe()` intrinsic halts the CPU when it enters the idle loop. Saving cycles and runtime energy

Performance analysis during hardware debugging

The code coverage and performance analysis tools are available when you are debugging on real hardware rather than simulation. However, to use these features you need two things: First, you need a microcontroller that has been fitted with the optional Embedded Trace Macrocell (ETM). Second, you need to use Keil ULINK pro debug adapter which supports instruction trace via the ETM.