Exercise 6 Time Management

In this exercise we will look at using the basic time osDelay() and delayUntil() functions

In the Pack Installer select “Ex 6 Time Management” and copy it to your tutorial directory.

This is our original led flasher program but the simple delay function has been replaced by the osDelay and osDelayUntil() API calls. LED2 is toggled every 100mS and LED1 is toggled every 500mS

```
void ledOn (void  *argument) {

 for (;;) {

        LED_On(1);

        osDelay(500);

        LED_Off(1);

        osDelay(500);

}}
```

In the Led2 thread we use the osDelayUntil() function to create a 1000 tick delay

```
__NO_RETURN void led2 (void  *argument) {

    for (;;) {

        ticks = osKernelGetTickCount();

          LED_On(2);

        osDelayUntil((ticks + 1000));          //Toggle LED 2 with an absolute delay

        LED_Off(2);

        osDelayUntil((ticks+2000));

    }

}
```

Build the project and start the debugger

Now we can see that the activity of the code is very different. When each of the LED tasks reaches the osDelay() API call it ‘blocks’ and moves to a waiting state. The appMain thread will be in a ready state so the scheduler will start it running. When the delay period has timed out the led tasks will move to the ready state and will be placed into the running state by the scheduler. This gives us a multi threaded program where CPU runtime is efficiently shared between threads.