COE718 - Lab 3b

Gabriel Casciano
500744076
Section: 7
gabriel.casciano@ryerson.ca

Oct, 28, 2019

Round Robin Scheduling

```
//Gabriel Casciano, 500744076
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <math.h>
#include "cmsis_os.h"
#include "RTL.H"
#include "LPC17xx.H"
#include "GLCD.h"

#define __FI        1
//#define __USE_LCD   0 //uncomment for DEMO


osThreadId idA, idB, idC, idD, idE;
int x, i, p, j, n, m;
char out[20];
#define PI 3.142

double factor;

__task void threadA (void const *arg) {
  int A = 0;
  for (x=0; x<257; x++){
    A = A + (x + (x+2));
    os_tsk_pass() ;//for concurrent execution
  }
#ifdef __USE_LCD
  GLCD_SetTextColor(Red);
  sprintf(out, "%d", (char)A);
  GLCD_DisplayString(4, 12, __FI, out);
#endif
  osThreadTerminate(idA);
}

__task void threadB (void const *arg) {

  int B = 0;
  factor=1;
  for(i = 1; i<17; i++){
    factor = factor*i;
    B = B + pow(2,i)/factor;
    os_tsk_pass() ;//for concurrent execution
  }
#ifdef __USE_LCD
  GLCD_SetTextColor(Red);
  sprintf(out, "%d", (char)B);
  GLCD_DisplayString(5, 12, __FI, out);
#endif
  osThreadTerminate(idB);
}

__task void threadC (void const *arg) {
  int C = 0;
  for (n=1; n<17; n++){
```

```
    C = C + (n+1)/n;
  }
#ifdef __USE_LCD
  GLCD_SetTextColor(Red);
  sprintf(out, "%d", (char)C);
  GLCD_DisplayString(6, 12, __FI, out);
#endif
  osThreadTerminate(idC);
}

__task void threadD (void const *arg) {
  int D = 0;
  factor=1;
  for (m=0; m<6; m++){
    factor = factor*m;
    if(factor == 0){
      factor=1;
    }
    else{
    os_tsk_pass() ;
    D = D + pow(5, m)/(double)factor;
    }
  }
#ifdef __USE_LCD
  GLCD_SetTextColor(Red);
  sprintf(out, "%d", (char)D);
  GLCD_DisplayString(7, 12, __FI, out);
#endif
  osThreadTerminate(idD);
}

__task void threadE(void const *arg) {
  int E = 0;
  int radius=1;
  for (p=1; p<13; p++){
    E = E + p*PI*(pow(radius, 2));
    os_tsk_pass() ;
  }
#ifdef __USE_LCD
  GLCD_SetTextColor(Red);
  sprintf(out, "%d", (char)E);
  GLCD_DisplayString(8, 12, __FI, out);
#endif
  osThreadTerminate(idE);
}

osThreadDef (threadA, osPriorityBelowNormal, 1, 0); //lowest priority
osThreadDef (threadB, osPriorityBelowNormal, 1, 0); //lowest priority
osThreadDef (threadC, osPriorityHigh, 1, 0); //highest Priority
osThreadDef (threadD, osPriorityAboveNormal, 1, 0); //medium priority
osThreadDef (threadE, osPriorityAboveNormal, 1, 0); //medium priority

int main (void) {
  SystemInit(); // initialize the Coretx-M3 processor

#ifdef __USE_LCD
  GLCD_Init();
  GLCD_Clear(White);
  GLCD_SetBackColor(Blue);
  GLCD_SetTextColor(Yellow);
  GLCD_DisplayString(0, 0, __FI, "  COE718 Demo Lab3b  ");
  GLCD_DisplayString(1, 0, __FI, "   Gabriel Casciano  ");
  GLCD_DisplayString(2, 0, __FI, "     500744076       ");
  GLCD_SetTextColor(White);
  GLCD_SetBackColor(White);
  GLCD_SetTextColor(Black);
  GLCD_DisplayString(4, 0, __FI, "Task A  :            ");
  GLCD_DisplayString(5, 0, __FI, "Task B  :            ");
  GLCD_DisplayString(6, 0, __FI, "Task C  :            ");
  GLCD_DisplayString(7, 0, __FI, "Task D  :            ");
  GLCD_DisplayString(8, 0, __FI, "Task E  :            ");
#endif

  osKernelInitialize ();   // setup kernel
  idA = osThreadCreate (osThread(threadA), NULL);   // create threads
  idB = osThreadCreate (osThread(threadB), NULL);
```

```
  idC = osThreadCreate (osThread(threadC), NULL);
  idD = osThreadCreate (osThread(threadD), NULL);
  idE = osThreadCreate (osThread(threadE), NULL);
  osKernelStart ();       // start kernel

  osDelay(osWaitForever);

}
```

Listing 1: Demo.c

```
/*----------------------------------------------------------------------------
 *      RL-ARM - RTX
 *----------------------------------------------------------------------------
 *      Name:    RTX_Conf_CM.C
 *      Purpose: Configuration of CMSIS RTX Kernel for Cortex-M
 *      Rev.:    V4.70
 *----------------------------------------------------------------------------
 *
 * Copyright (c) 1999-2009 KEIL, 2009-2013 ARM Germany GmbH
 * All rights reserved.
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *  - Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *  - Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *  - Neither the name of ARM  nor the names of its contributors may be used
 *    to endorse or promote products derived from this software without
 *    specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL COPYRIGHT HOLDERS AND CONTRIBUTORS BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 *----------------------------------------------------------------------------*/

#include "cmsis_os.h"

unsigned int countIDLE = 0;

/*----------------------------------------------------------------------------
 *      RTX User configuration part BEGIN
 *----------------------------------------------------------------------------*/

//-------- <<< Use Configuration Wizard in Context Menu >>> -----------------
//
// <h>Thread Configuration
// =======================
//
//   <o>Number of concurrent running threads <0-250>
//   <i> Defines max. number of threads that will run at the same time.
//   <i> Default: 6
#ifndef OS_TASKCNT
 #define OS_TASKCNT     6
#endif

//   <o>Default Thread stack size [bytes] <64-4096:8><#/4>
//   <i> Defines default stack size for threads with osThreadDef stacksz = 0
//   <i> Default: 200
#ifndef OS_STKSIZE
 #define OS_STKSIZE     50
#endif

//   <o>Main Thread stack size [bytes] <64-4096:8><#/4>
//   <i> Defines stack size for main thread.
//   <i> Default: 200
```

```
#ifndef OS_MAINSTKSIZE
 #define OS_MAINSTKSIZE 50
#endif

//   <o>Number of threads with user-provided stack size <0-250>
//   <i> Defines the number of threads with user-provided stack size.
//   <i> Default: 0
#ifndef OS_PRIVCNT
 #define OS_PRIVCNT     0
#endif

//   <o>Total stack size [bytes] for threads with user-provided stack size <0-4096:8><#/4>
//   <i> Defines the combined stack size for threads with user-provided stack size.
//   <i> Default: 0
#ifndef OS_PRIVSTKSIZE
 #define OS_PRIVSTKSIZE 0
#endif

// <q>Check for stack overflow
// <i> Includes the stack checking code for stack overflow.
// <i> Note that additional code reduces the Kernel performance.
#ifndef OS_STKCHECK
 #define OS_STKCHECK    1
#endif

// <o>Processor mode for thread execution
//   <0=> Unprivileged mode
//   <1=> Privileged mode
// <i> Default: Privileged mode
#ifndef OS_RUNPRIV
 #define OS_RUNPRIV     0
#endif

// </h>

// <h>RTX Kernel Timer Tick Configuration
// ======================================
// <q> Use Cortex-M SysTick timer as RTX Kernel Timer
// <i> Use the Cortex-M SysTick timer as a time-base for RTX.
#ifndef OS_SYSTICK
 #define OS_SYSTICK     1
#endif
//
//   <o>Timer clock value [Hz] <1-1000000000>
//   <i> Defines the timer clock value.
//   <i> Default: 12000000  (12MHz)
#ifndef OS_CLOCK
 #define OS_CLOCK       10000000
#endif

//   <o>Timer tick value [us] <1-1000000>
//   <i> Defines the timer tick value.
//   <i> Default: 1000  (1ms)
#ifndef OS_TICK
 #define OS_TICK        10000
#endif

// </h>

// <h>System Configuration
// =======================
//
// <e>Round-Robin Thread switching
// ===============================
//
// <i> Enables Round-Robin Thread switching.
#ifndef OS_ROBIN
 #define OS_ROBIN       1
#endif

//   <o>Round-Robin Timeout [ticks] <1-1000>
//   <i> Defines how long a thread will execute before a thread switch.
//   <i> Default: 5
#ifndef OS_ROBINTOUT
 #define OS_ROBINTOUT   10
#endif
```

```
141
142 // </e>
143
144 // <e>User Timers
145 // ==============
146 //   <i> Enables user Timers
147 #ifndef OS_TIMERS
148  #define OS_TIMERS      1
149 #endif
150
151 //   <o>Timer Thread Priority
152 //                        <1=> Low
153 //     <2=> Below Normal  <3=> Normal  <4=> Above Normal
154 //                        <5=> High
155 //                        <6=> Realtime (highest)
156 //   <i> Defines priority for Timer Thread
157 //   <i> Default: High
158 #ifndef OS_TIMERPRIO
159  #define OS_TIMERPRIO   5
160 #endif
161
162 //   <o>Timer Thread stack size [bytes] <64-4096:8><#/4>
163 //   <i> Defines stack size for Timer thread.
164 //   <i> Default: 200
165 #ifndef OS_TIMERSTKSZ
166  #define OS_TIMERSTKSZ  50
167 #endif
168
169 //   <o>Timer Callback Queue size <1-32>
170 //   <i> Number of concurrent active timer callback functions.
171 //   <i> Default: 4
172 #ifndef OS_TIMERCBQS
173  #define OS_TIMERCBQS   4
174 #endif
175
176 // </e>
177
178 //   <o>ISR FIFO Queue size<4=>   4 entries  <8=>   8 entries
179 //                          <12=> 12 entries  <16=> 16 entries
180 //                          <24=> 24 entries  <32=> 32 entries
181 //                          <48=> 48 entries  <64=> 64 entries
182 //                          <96=> 96 entries
183 //   <i> ISR functions store requests to this buffer,
184 //   <i> when they are called from the interrupt handler.
185 //   <i> Default: 16 entries
186 #ifndef OS_FIFOSZ
187  #define OS_FIFOSZ      16
188 #endif
189
190 // </h>
191
192 //------------- <<< end of configuration section >>> -----------------------
193
194 // Standard library system mutexes
195 // ===============================
196 //  Define max. number system mutexes that are used to protect
197 //  the arm standard runtime library. For microlib they are not used.
198 #ifndef OS_MUTEXCNT
199  #define OS_MUTEXCNT    8
200 #endif
201
202 /*----------------------------------------------------------------------------
203  *      RTX User configuration part END
204  *---------------------------------------------------------------------------*/
205
206 #define OS_TRV          ((uint32_t)(((double)OS_CLOCK*(double)OS_TICK)/1E6)-1)
207
208
209 /*----------------------------------------------------------------------------
210  *      Global Functions
211  *---------------------------------------------------------------------------*/
212
213 /*--------------------------- os_idle_demon ---------------------------------*/
214
215 void os_idle_demon (void) {
216   /* The idle demon is a system thread, running when no other thread is      */
```

```
217   /* ready to run.                                                      */
218
219   for (;;) {
220     /* HERE: include optional user code to be executed when no thread runs.*/
221   }
222 }
223
224 #if (OS_SYSTICK == 0)   // Functions for alternative timer as RTX kernel timer
225
226 /*---------------------------- os_tick_init -----------------------------------*/
227
228 // Initialize alternative hardware timer as RTX kernel timer
229 // Return: IRQ number of the alternative hardware timer
230 int os_tick_init (void) {
231   return (-1);  /* Return IRQ number of timer (0..239) */
232 }
233
234 /*---------------------------- os_tick_val ------------------------------------*/
235
236 // Get alternative hardware timer current value (0 .. OS_TRV)
237 uint32_t os_tick_val (void) {
238   return (0);
239 }
240
241 /*---------------------------- os_tick_ovf ------------------------------------*/
242
243 // Get alternative hardware timer overflow flag
244 // Return: 1 - overflow, 0 - no overflow
245 uint32_t os_tick_ovf (void) {
246   return (0);
247 }
248
249 /*---------------------------- os_tick_irqack ---------------------------------*/
250
251 // Acknowledge alternative hardware timer interrupt
252 void os_tick_irqack (void) {
253   /* ... */
254 }
255
256 #endif   // (OS_SYSTICK == 0)
257
258 /*---------------------------- os_error ---------------------------------------*/
259
260 void os_error (uint32_t err_code) {
261   /* This function is called when a runtime error is detected. Parameter */
262   /* 'err_code' holds the runtime error code (defined in RTL.H).         */
263
264   /* HERE: include optional code to be executed on runtime error. */
265   for (;;);
266 }
267
268
269 /*----------------------------------------------------------------------------
270  *      RTX Configuration Functions
271  *---------------------------------------------------------------------------*/
272
273 #include "RTX_CM_lib.h"
274
275 /*----------------------------------------------------------------------------
276  * end of file
277  *---------------------------------------------------------------------------*/
```

Listing 2: RTX_Conf_CM.c

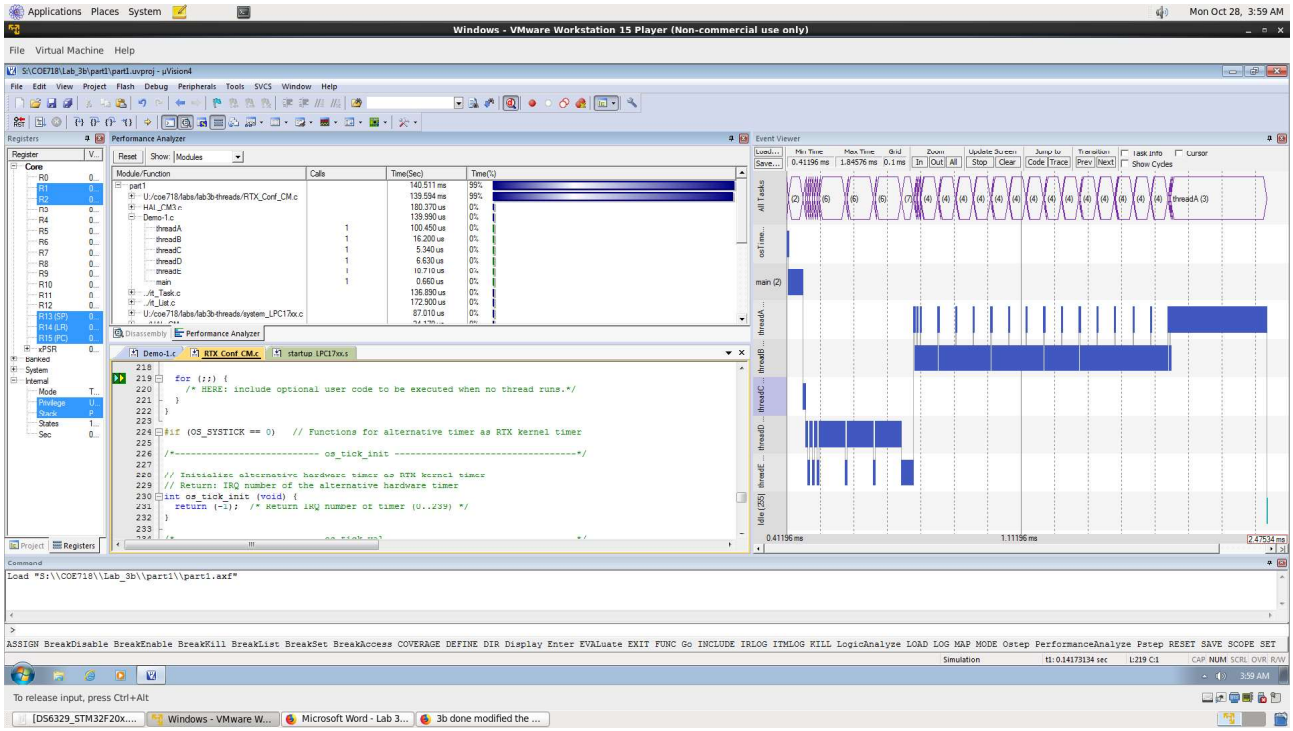

Figure 1: Task Timing and Performance Analyzer

Preemptive Scheduling

```c
//Gabriel Casciano, 500744076
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <math.h>
#include "cmsis_os.h"
#include "RTL.H"
#include "LPC17xx.H"
#include "GLCD.h"

#define __FI        1
//#define __USE_LCD   0 //uncomment for DEMO


osThreadId idA, idB, idc, idD, idE;

#define PI 3.142

double factor;

__task void threadA (void const *arg) {
 char out[20];
 int A = 0;
 int x = 0;
 for ( x=0; x<257; x++){
  A = A + (x + (x+2));
  os_tsk_pass() ;//for concurrent execution
 }
#ifdef __USE_LCD
 GLCD_SetTextColor(Red);
 sprintf(out, "%d", A);
 GLCD_DisplayString(4, 12, __FI, (unsigned char*) out);
#endif
 osThreadTerminate(idA);
}

__task void threadB (void const *arg) {
  float B = 0;
  char out[20];

 int i = 0;
```

```
 factor=1;
 for( i = 1; i<17; i++){
  factor = factor*i;
  B = B + pow(2,i)/factor;
  os_tsk_pass() ;//for concurrent execution
 }
#ifdef __USE_LCD
 GLCD_SetTextColor(Red);
 sprintf(out, "%f", B);
 GLCD_DisplayString(5, 12, __FI, (unsigned char*) out);
#endif
 osThreadTerminate(idB);
}

__task void threadC (void const *arg) {
 float C = 0;
  char out[20];

 int n=0;
 for ( n=1; n<17; n++){
  C = C + (n+1)/n;
 }
#ifdef __USE_LCD

 GLCD_SetTextColor(Red);
 sprintf(out, "%f", C);
 GLCD_DisplayString(6, 12, __FI, (unsigned char*) out);
#endif
 osThreadTerminate(idc);
}

__task void threadD (void const *arg) {
 float D = 0;
  char out[20];

 int m=0;
 factor=1;
 for ( m=0; m<6; m++){
  factor = factor*m;
  if(factor == 0){
   factor=1;
  }
  else{
  os_tsk_pass() ;
  D = D + pow(5, m)/(double)factor;
  }
 }
#ifdef __USE_LCD
 GLCD_SetTextColor(Red);
 sprintf(out, "%f", D);
 GLCD_DisplayString(7, 12, __FI, (unsigned char*) out);
#endif
 osThreadTerminate(idD);
}

__task void threadE(void const *arg) {
 int E = 0;
 char out[20];

 int p=0;
 int radius=1;
 for (p=1; p<13; p++){
  E = E + p*PI*(pow(radius, 2));
  os_tsk_pass() ;
 }
#ifdef __USE_LCD
 GLCD_SetTextColor(Red);
 sprintf(out, "%d", E);
 GLCD_DisplayString(8, 12, __FI, (unsigned char*) out);
#endif
 osThreadTerminate(idE);
}

osThreadDef (threadA, osPriorityAboveNormal, 1, 0); //lowest priority
osThreadDef (threadB, osPriorityBelowNormal, 1, 0); //lowest priority
osThreadDef (threadC, osPriorityHigh, 1, 0); //highest Priority
```

```
osThreadDef (threadD, osPriorityAboveNormal, 1, 0); //medium priority
osThreadDef (threadE, osPriorityBelowNormal, 1, 0); //medium priority

int main (void) {
 SystemInit(); // initialize the Coretx-M3 processor

#ifdef __USE_LCD
 GLCD_Init();
 GLCD_Clear(White);
 GLCD_SetBackColor(Blue);
 GLCD_SetTextColor(Yellow);
 GLCD_DisplayString(0, 0, __FI, "  COE718 Demo Lab3b  ");
 GLCD_DisplayString(1, 0, __FI, "   Gabriel Casciano  ");
 GLCD_DisplayString(2, 0, __FI, "     500744076       ");
 GLCD_SetTextColor(White);
 GLCD_SetBackColor(White);
 GLCD_SetTextColor(Black);
 GLCD_DisplayString(4, 0, __FI, "Task A  :            ");
 GLCD_DisplayString(5, 0, __FI, "Task B  :            ");
 GLCD_DisplayString(6, 0, __FI, "Task C  :            ");
 GLCD_DisplayString(7, 0, __FI, "Task D  :            ");
 GLCD_DisplayString(8, 0, __FI, "Task E  :            ");
#endif

 osKernelInitialize ();   // setup kernel
 idA = osThreadCreate (osThread(threadA), NULL);   // create threads
  idB = osThreadCreate (osThread(threadB), NULL);
 idc = osThreadCreate (osThread(threadC), NULL);
 idD = osThreadCreate (osThread(threadD), NULL);
 idE = osThreadCreate (osThread(threadE), NULL);
  osKernelStart ();       // start kernel

 osDelay(osWaitForever);

}
```

Listing 3: Demo2.c

```
/*----------------------------------------------------------------------------
 *      RL-ARM - RTX
 *----------------------------------------------------------------------------
 *      Name:    RTX_Conf_CM.C
 *      Purpose: Configuration of CMSIS RTX Kernel for Cortex-M
 *      Rev.:    V4.70
 *----------------------------------------------------------------------------
 *
 * Copyright (c) 1999-2009 KEIL, 2009-2013 ARM Germany GmbH
 * All rights reserved.
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *  - Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *  - Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *  - Neither the name of ARM  nor the names of its contributors may be used
 *    to endorse or promote products derived from this software without
 *    specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL COPYRIGHT HOLDERS AND CONTRIBUTORS BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 *----------------------------------------------------------------------------*/

#include "cmsis_os.h"

unsigned int countIDLE = 0;
```

```
38
39 /*----------------------------------------------------------------------------
40  *      RTX User configuration part BEGIN
41  *---------------------------------------------------------------------------*/
42
43 //-------- <<< Use Configuration Wizard in Context Menu >>> -----------------
44 //
45 // <h>Thread Configuration
46 // =======================
47 //
48 //   <o>Number of concurrent running threads <0-250>
49 //   <i> Defines max. number of threads that will run at the same time.
50 //   <i> Default: 6
51 #ifndef OS_TASKCNT
52  #define OS_TASKCNT     6
53 #endif
54
55 //   <o>Default Thread stack size [bytes] <64-4096:8><#/4>
56 //   <i> Defines default stack size for threads with osThreadDef stacksz = 0
57 //   <i> Default: 200
58 #ifndef OS_STKSIZE
59  #define OS_STKSIZE     50
60 #endif
61
62 //   <o>Main Thread stack size [bytes] <64-4096:8><#/4>
63 //   <i> Defines stack size for main thread.
64 //   <i> Default: 200
65 #ifndef OS_MAINSTKSIZE
66  #define OS_MAINSTKSIZE 50
67 #endif
68
69 //   <o>Number of threads with user-provided stack size <0-250>
70 //   <i> Defines the number of threads with user-provided stack size.
71 //   <i> Default: 0
72 #ifndef OS_PRIVCNT
73  #define OS_PRIVCNT     0
74 #endif
75
76 //   <o>Total stack size [bytes] for threads with user-provided stack size <0-4096:8><#/4>
77 //   <i> Defines the combined stack size for threads with user-provided stack size.
78 //   <i> Default: 0
79 #ifndef OS_PRIVSTKSIZE
80  #define OS_PRIVSTKSIZE 0
81 #endif
82
83 // <q>Check for stack overflow
84 // <i> Includes the stack checking code for stack overflow.
85 // <i> Note that additional code reduces the Kernel performance.
86 #ifndef OS_STKCHECK
87  #define OS_STKCHECK    1
88 #endif
89
90 // <o>Processor mode for thread execution
91 //   <0=> Unprivileged mode
92 //   <1=> Privileged mode
93 // <i> Default: Privileged mode
94 #ifndef OS_RUNPRIV
95  #define OS_RUNPRIV     0
96 #endif
97
98 // </h>
99
100 // <h>RTX Kernel Timer Tick Configuration
101 // ======================================
102 // <q> Use Cortex-M SysTick timer as RTX Kernel Timer
103 // <i> Use the Cortex-M SysTick timer as a time-base for RTX.
104 #ifndef OS_SYSTICK
105  #define OS_SYSTICK     1
106 #endif
107 //
108 //   <o>Timer clock value [Hz] <1-1000000000>
109 //   <i> Defines the timer clock value.
110 //   <i> Default: 12000000  (12MHz)
111 #ifndef OS_CLOCK
112  #define OS_CLOCK       10000000
113 #endif
```

```
114
115 //   <o>Timer tick value [us] <1-1000000>
116 //   <i> Defines the timer tick value.
117 //   <i> Default: 1000  (1ms)
118 #ifndef OS_TICK
119  #define OS_TICK        10000
120 #endif
121
122 // </h>
123
124 // <h>System Configuration
125 // =======================
126 //
127 // <e>Round-Robin Thread switching
128 // ===============================
129 //
130 // <i> Enables Round-Robin Thread switching.
131 #ifndef OS_ROBIN
132  #define OS_ROBIN       0
133 #endif
134
135 //   <o>Round-Robin Timeout [ticks] <1-1000>
136 //   <i> Defines how long a thread will execute before a thread switch.
137 //   <i> Default: 5
138 #ifndef OS_ROBINTOUT
139  #define OS_ROBINTOUT   10
140 #endif
141
142 // </e>
143
144 // <e>User Timers
145 // ==============
146 //   <i> Enables user Timers
147 #ifndef OS_TIMERS
148  #define OS_TIMERS      1
149 #endif
150
151 //   <o>Timer Thread Priority
152 //                        <1=> Low
153 //     <2=> Below Normal  <3=> Normal  <4=> Above Normal
154 //                        <5=> High
155 //                        <6=> Realtime (highest)
156 //   <i> Defines priority for Timer Thread
157 //   <i> Default: High
158 #ifndef OS_TIMERPRIO
159  #define OS_TIMERPRIO   5
160 #endif
161
162 //   <o>Timer Thread stack size [bytes] <64-4096:8><#/4>
163 //   <i> Defines stack size for Timer thread.
164 //   <i> Default: 200
165 #ifndef OS_TIMERSTKSZ
166  #define OS_TIMERSTKSZ  50
167 #endif
168
169 //   <o>Timer Callback Queue size <1-32>
170 //   <i> Number of concurrent active timer callback functions.
171 //   <i> Default: 4
172 #ifndef OS_TIMERCBQS
173  #define OS_TIMERCBQS   4
174 #endif
175
176 // </e>
177
178 //   <o>ISR FIFO Queue size<4=>   4 entries  <8=>   8 entries
179 //                         <12=> 12 entries  <16=> 16 entries
180 //                         <24=> 24 entries  <32=> 32 entries
181 //                         <48=> 48 entries  <64=> 64 entries
182 //                         <96=> 96 entries
183 //   <i> ISR functions store requests to this buffer,
184 //   <i> when they are called from the interrupt handler.
185 //   <i> Default: 16 entries
186 #ifndef OS_FIFOSZ
187  #define OS_FIFOSZ      16
188 #endif
189
```

```
// </h>

//------------- <<< end of configuration section >>> ----------------------

// Standard library system mutexes
// ===============================
//  Define max. number system mutexes that are used to protect
//  the arm standard runtime library. For microlib they are not used.
#ifndef OS_MUTEXCNT
 #define OS_MUTEXCNT    8
#endif

/*----------------------------------------------------------------------------
 *      RTX User configuration part END
 *---------------------------------------------------------------------------*/

#define OS_TRV          ((uint32_t)(((double)OS_CLOCK*(double)OS_TICK)/1E6)-1)


/*----------------------------------------------------------------------------
 *      Global Functions
 *---------------------------------------------------------------------------*/

/*--------------------------- os_idle_demon ---------------------------------*/

void os_idle_demon (void) {
  /* The idle demon is a system thread, running when no other thread is      */
  /* ready to run.                                                           */

  for (;;) {
    /* HERE: include optional user code to be executed when no thread runs.*/
  }
}

#if (OS_SYSTICK == 0)   // Functions for alternative timer as RTX kernel timer

/*--------------------------- os_tick_init ----------------------------------*/

// Initialize alternative hardware timer as RTX kernel timer
// Return: IRQ number of the alternative hardware timer
int os_tick_init (void) {
  return (-1);  /* Return IRQ number of timer (0..239) */
}

/*--------------------------- os_tick_val -----------------------------------*/

// Get alternative hardware timer current value (0 .. OS_TRV)
uint32_t os_tick_val (void) {
  return (0);
}

/*--------------------------- os_tick_ovf -----------------------------------*/

// Get alternative hardware timer overflow flag
// Return: 1 - overflow, 0 - no overflow
uint32_t os_tick_ovf (void) {
  return (0);
}

/*--------------------------- os_tick_irqack --------------------------------*/

// Acknowledge alternative hardware timer interrupt
void os_tick_irqack (void) {
  /* ... */
}

#endif   // (OS_SYSTICK == 0)

/*--------------------------- os_error --------------------------------------*/

void os_error (uint32_t err_code) {
  /* This function is called when a runtime error is detected. Parameter */
  /* 'err_code' holds the runtime error code (defined in RTL.H).         */

  /* HERE: include optional code to be executed on runtime error. */
  for (;;);
```

```
266 }
267
268
269 /*----------------------------------------------------------------------------
270  *       RTX Configuration Functions
271  *---------------------------------------------------------------------------*/
272
273 #include "RTX_CM_lib.h"
274
275 /*----------------------------------------------------------------------------
276  * end of file
277  *---------------------------------------------------------------------------*/
```

Listing 4: RTX_Conf_CM.c

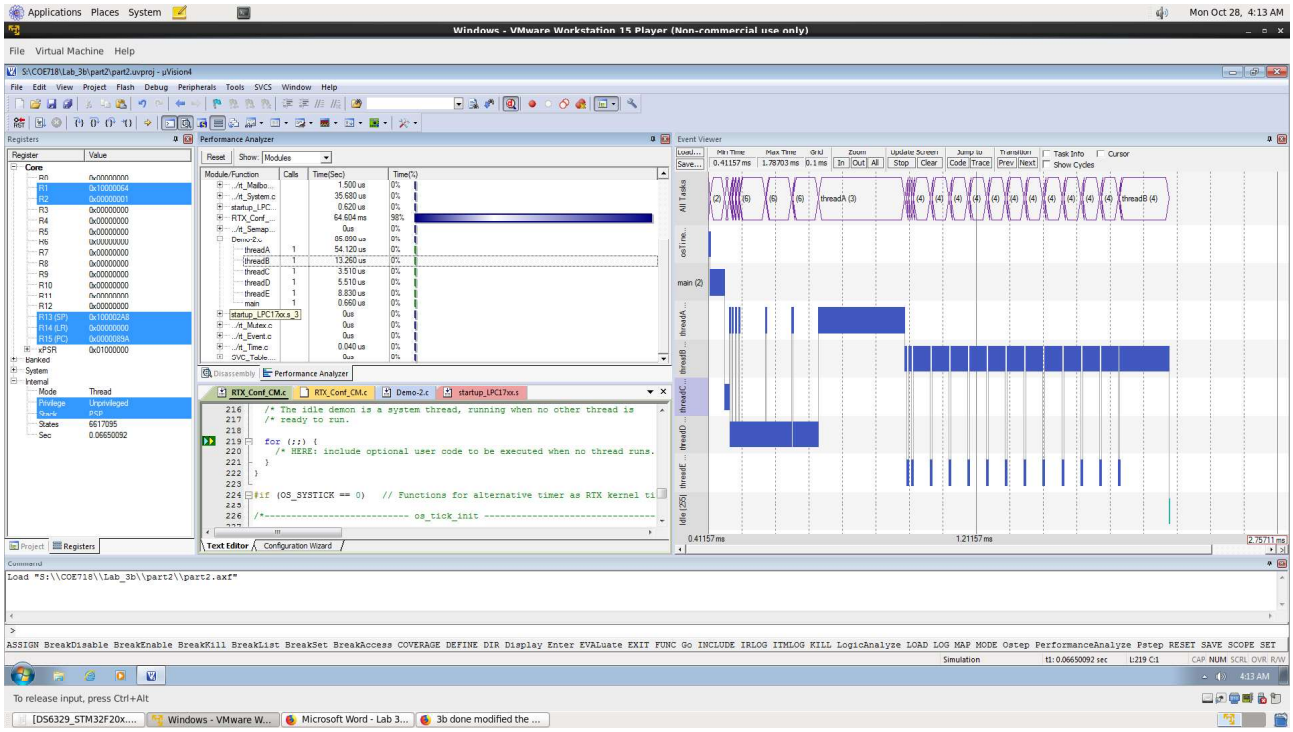

Figure 2: Task Timing and Performance Analyzer