COE718 - Lab 3a

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

Oct, 28, 2019

Round Robin Scheduling

```c
//Gabriel Casciano, 500744076
#include <stdio.h>
#include "LPC17xx.h"
#include <RTL.h>
#include "GLCD.h"
#include "LED.h"

#define __FI        1                      /* Font index 16x24         */
//#define __USE_LCD   0

long global_c1 = 0, global_c2 = 0, global_c3 = 0;


__task void task1(void){
#ifdef __USE_LCD

  GLCD_SetTextColor(DarkGrey);
  GLCD_DisplayString(1, 1, __FI, "Appetizer");
  GLCD_DisplayString(4, 1, __FI, "Entree");
  GLCD_DisplayString(7, 1, __FI, "Dessert");
#endif
  global_c1 = 0xFEEED;
  while(global_c1>0){
    global_c1-=2;
    LED_Out(0x81);
  }
#ifdef __USE_LCD
  GLCD_SetTextColor(Blue);
  GLCD_DisplayString(1, 12, __FI, "TASTY!");
#endif
  os_tsk_delete_self();
}

__task void task2(void){
  global_c2 = 0xFEEED;
  while(global_c2>0){
    global_c2-=3;
    LED_Out(0x24);
  }
#ifdef __USE_LCD
  GLCD_SetTextColor(Red);
  GLCD_DisplayString(4, 12, __FI, "YUUUM!");
  os_tsk_delete_self();
#endif
}

__task void task3(void){
  global_c3 = 0xFEEED;
  while(global_c3>0){
    global_c3 -= 4;
    LED_Out(0x18);
  }
#ifdef __USE_LCD
  GLCD_DisplayString(7, 10, __FI, "DELICIOUS!");
  os_tsk_delete_self();
#endif
}

int main(void){
#ifdef __USE_LCD
  GLCD_Init();
  GLCD_Clear(White);
#endif
  SystemInit();
  LED_Init();
  os_tsk_create(task1, 1);
  os_tsk_create(task2, 1);
  os_tsk_create(task3, 1);

  os_tsk_delete_self();

  os_sys_init(task1);
}
```

Listing 1: Demo.c

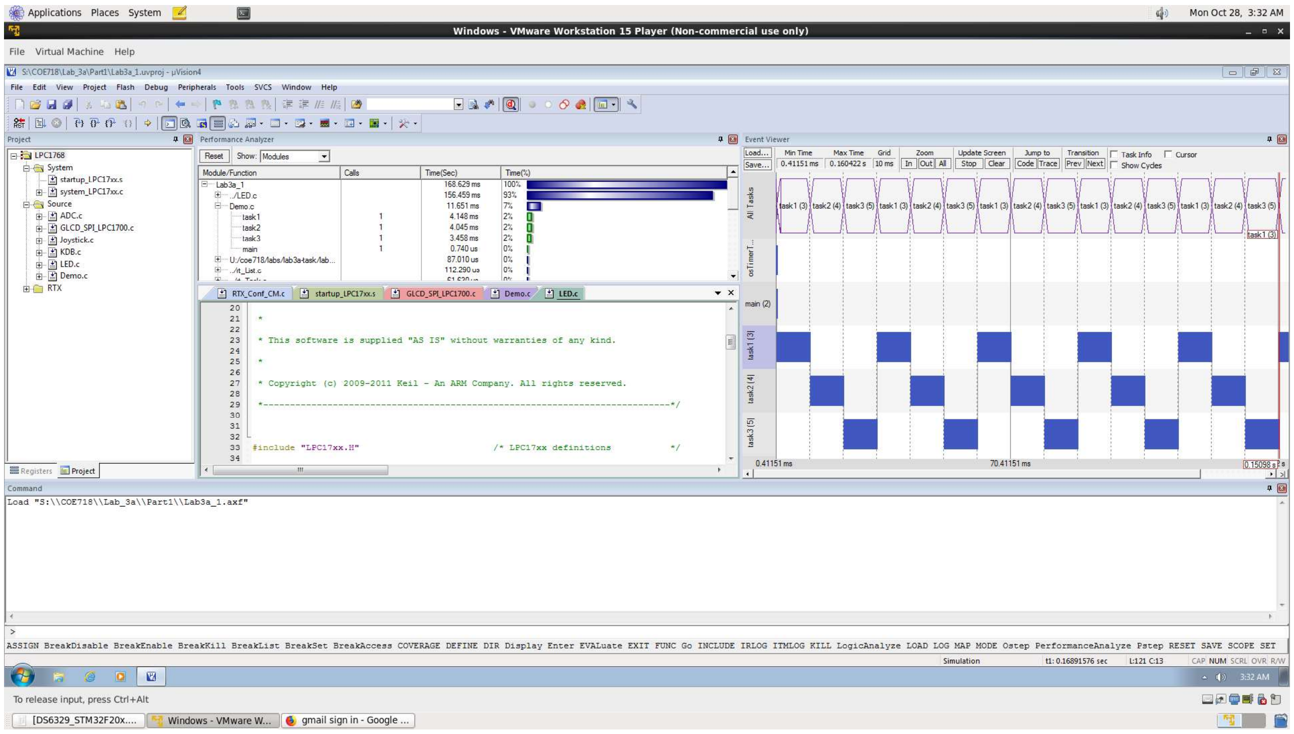
```c
/*----------------------------------------------------------------------------
 *      RL-ARM - RTX
 *----------------------------------------------------------------------------
 *      Name:    RTX_Conf_CM.C
 *      Purpose: Configuration of CMSIS RTX Kernel for Cortex-M
 *      Rev.:    V4.70
 *----------------------------------------------------------------------------
 *
 * 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

// </e>

// <e>User Timers
// ==============
//   <i> Enables user Timers
#ifndef OS_TIMERS
 #define OS_TIMERS      1
#endif

//   <o>Timer Thread Priority
//                        <1=> Low
//     <2=> Below Normal  <3=> Normal  <4=> Above Normal
//                        <5=> High
//                        <6=> Realtime (highest)
//   <i> Defines priority for Timer Thread
//   <i> Default: High
#ifndef OS_TIMERPRIO
 #define OS_TIMERPRIO   5
#endif

//   <o>Timer Thread stack size [bytes] <64-4096:8><#/4>
//   <i> Defines stack size for Timer thread.
//   <i> Default: 200
#ifndef OS_TIMERSTKSZ
 #define OS_TIMERSTKSZ  50
#endif

//   <o>Timer Callback Queue size <1-32>
//   <i> Number of concurrent active timer callback functions.
//   <i> Default: 4
#ifndef OS_TIMERCBQS
 #define OS_TIMERCBQS   4
#endif

// </e>

//   <o>ISR FIFO Queue size<4=>   4 entries  <8=>   8 entries
//                         <12=> 12 entries  <16=> 16 entries
//                         <24=> 24 entries  <32=> 32 entries
//                         <48=> 48 entries  <64=> 64 entries
//                         <96=> 96 entries
//   <i> ISR functions store requests to this buffer,
//   <i> when they are called from the interrupt handler.
//   <i> Default: 16 entries
#ifndef OS_FIFOSZ
 #define OS_FIFOSZ      16
#endif

// </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)
```

```c


/*----------------------------------------------------------------------------
 *      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 (;;);
}


/*----------------------------------------------------------------------------
 *      RTX Configuration Functions
 *---------------------------------------------------------------------------*/

#include "RTX_CM_lib.h"

/*----------------------------------------------------------------------------
 * end of file
 *---------------------------------------------------------------------------*/
```

Listing 2: RTX_Conf_CM.c



Module/Function	Calls	Time(Sec)	Time(%)
Lab3a_1		168.629 ms	100%
../LED.c		156.459 ms	93%
Demo.c		11.651 ms	7%
task1	1	4.148 ms	2%
task2	1	4.045 ms	2%
task3	1	3.458 ms	2%
main	1	0.740 us	0%
U:/coe718/labs/lab3a-task/lab...		87.010 us	0%
../rt_List.c		112.290 us	0%

```
 *
 * This software is supplied "AS IS" without warranties of any kind.
 *
 * Copyright (c) 2009-2011 Keil - An ARM Company. All rights reserved.
 *---------------------------------------------------------------------------*/


#include "LPC17xx.H"                        /* LPC17xx definitions          */
```

```
Load "S:\\COE718\\Lab_3a\\Part1\\Lab3a_1.axf"
```

```
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALuate EXIT FUNC Go INCLUDE IRLOG ITMLOG KILL LogicAnalyze LOAD LOG MAP MODE Ostep PerformanceAnalyze Pstep RESET SAVE SCOPE SET
```

Figure 1: Task Timing and Performance Analyzer

Preemptive Scheduling

```c
//Gabriel Casciano, 500744076
#include <stdio.h>
#include "LPC17xx.h"
#include <RTL.h>
#include "GLCD.h"
#include "LED.h"
#include <string.h>

#define __FI        1                      /* Font index 16x24         */
//#define __USE_LCD   0

OS_TID MEMid, CPUid, APPid, DEVid, id5;
double  MEMcnt, CPUcnt, APPcnt, DEVcnt, users;
OS_MUT mutex;
char logger[];

// Bit Band Macros used to calculate the alias address at run time
#define ADDRESS(x)    (*((volatile unsigned long *)(x)))
#define BitBand(x, y)   ADDRESS(((unsigned long)(x) & 0xF0000000) | 0x02000000 |(((unsigned
    long)(x) & 0x000FFFFF) << 5) | ((y) << 2))
#define L1 (*((volatile unsigned long *)0x233806F0))        //assigns LED on port1.28 for
    bitbanding

int r1 = 1, r2 = 0, r3 = 5;

int i,j;
void delay(){
  for(i = 0 ; i < 10000000; i++){
    j=i;
  }
}

__task void MemoryM (void);
__task void CPUM (void);
__task void AppI (void);
__task void DeviceM (void);
__task void UserI (void);


__task void MemoryM (void) {

```

```
#ifdef __USE_LCD
  GLCD_SetTextColor(Magenta);
  GLCD_DisplayString(7, 1, __FI, "Memory Management");
  LED_Out(1);
  delay();
#endif

  MEMid = os_tsk_self();  //identify myself and create CPU management
  os_tsk_pass();                  //passes control to CPU management
  MEMcnt++;                       //increment counter
  L1 = 1;                        //bitbanding




  if(os_evt_wait_and(0x0004, 0xFFFF)){    //receives signal back from CPUm
    L1 = 0;   //bitbanding switch port 1.28 off
    //delay();
    os_tsk_delete_self(); //delete itself (MemoryM)
  }
}

__task void CPUM (void) {

#ifdef __USE_LCD
  GLCD_SetTextColor(Magenta);
  GLCD_DisplayString(7, 1, __FI, "CPU Management    ");
  LED_Out(2);
  delay();
#endif

  CPUid = os_tsk_self(); //obtain my identity
  //os_tsk_pass(); //pass to Memory M
    // barrel-shifter & conditional execution
    while(r2 <= 0x18){
      if((r1 - r2) > 0){
        r1 = r1 + 2;
        r2 = r1 + (r3*4);
        r3 = r3/2;
      }
      else{
        r2 = r2 + 1;
      }
    }
    CPUcnt++; //increment counter
    //delay();
    os_evt_set(0X0004,MEMid); //signals back to memory management
    os_tsk_delete_self();
}


__task void AppI (void) {

#ifdef __USE_LCD
  GLCD_SetTextColor(Magenta);
  GLCD_DisplayString(7, 1, __FI, "App Interface   ");
  LED_Out(4);
  delay();
#endif
  APPid = os_tsk_self();              //obtain my identity
  os_mut_init(mutex);               //initialization of the system
  os_mut_wait(&mutex, 0xffff);      // in the task seeking mutual exclusion
  strcpy(logger,"Start-");
  os_tsk_pass(); //passing token to Device Management
  //os_tsk_prio_self(7);  //increase my priority so that I may get Device Management's signal
  if(os_evt_wait_and(0x0008, 0xFFFF)){    //receives signal back from DeviceM
    APPcnt++;     //increment counter
    //delay();
    os_tsk_delete(DEVid);   //delete device manager
  }
}

__task void DeviceM (void) {
#ifdef __USE_LCD
  GLCD_SetTextColor(Magenta);
  GLCD_DisplayString(7, 1, __FI, "Device Manager  ");
```

```c
  LED_Out(8);
  delay();
#endif

  DEVid = os_tsk_self(); //obtain my identity
  os_evt_set(0X0008,APPid); //signals back to App Interface
  os_tsk_pass();  //pass to App Interface so it executes before Device Management
  strcpy(logger,"End");
  DEVcnt++; //increment counter
  //delay();
}

__task void UserI (void) {

#ifdef __USE_LCD
  GLCD_SetTextColor(Magenta);
  GLCD_DisplayString(7, 1, __FI, "User Interface  ");
  LED_Out(16);
  delay();
#endif
  users++; //increment users
  //delay();
  os_tsk_delete_self();
}

int main (void) {
  LED_Init();                                /* LED Initialization            */

#ifdef __USE_LCD
  GLCD_Init();                                /* Initialize graphical LCD (if enabled  */

  GLCD_Clear(White);                         /* Clear graphical LCD display   */
  GLCD_SetBackColor(Black);
  GLCD_SetTextColor(Yellow);
  GLCD_DisplayString(0, 0, __FI, "Anne's COE718 Demo     ");
  GLCD_SetTextColor(White);
  GLCD_DisplayString(1, 0, __FI, "       Demo2.c      ");
  GLCD_DisplayString(2, 0, __FI, "Preemptive Scheduling");
  GLCD_SetBackColor(White);
  GLCD_SetTextColor(DarkCyan);
  GLCD_DisplayString(5, 0, __FI, "Task:          ");
#endif
  os_tsk_create(MemoryM, 1); //create MemoryManagement and initialize system
  os_tsk_create(CPUM, 1);
  os_tsk_create(DeviceM, 2);
  os_tsk_create(AppI, 2);
  os_tsk_create(UserI, 5);
  SystemInit();
  os_mut_init(&mutex);
  os_tsk_delete_self();
}
```

Listing 3: Demo2.c

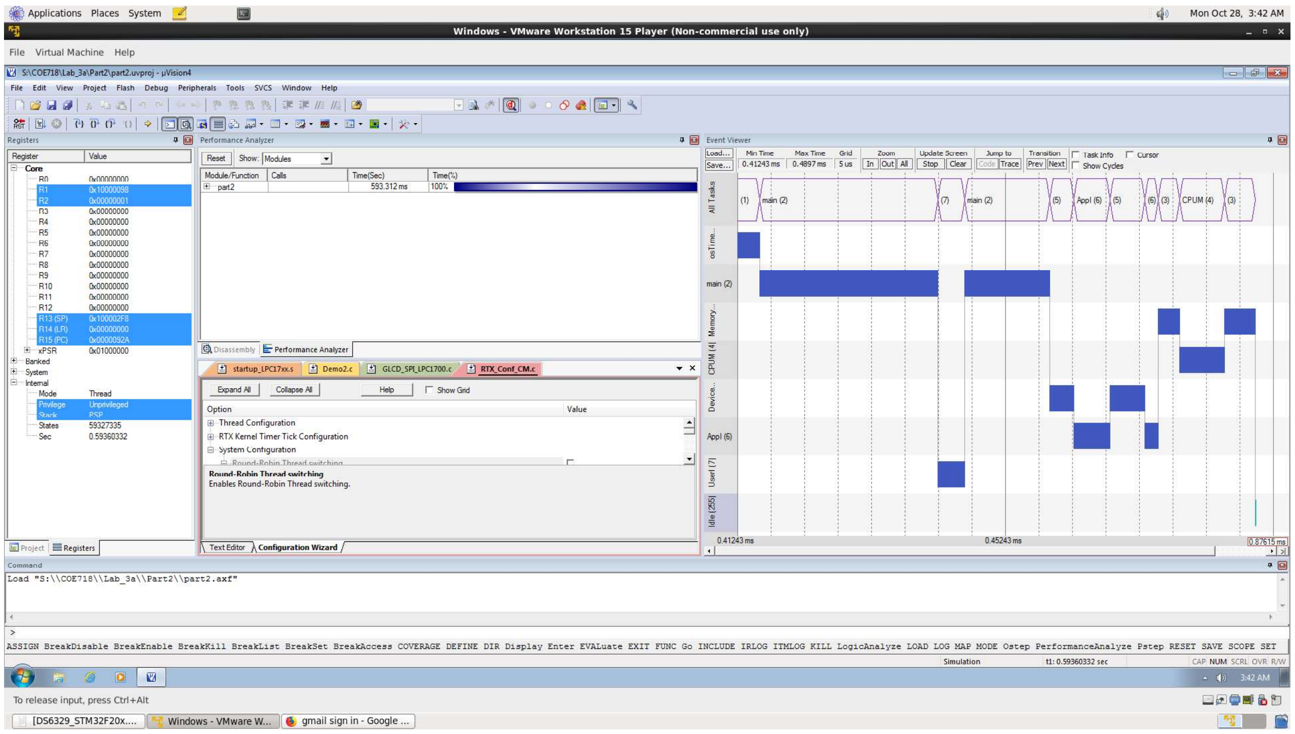
```c
/*----------------------------------------------------------------------------
 *      RL-ARM - RTX
 *----------------------------------------------------------------------------
 *      Name:    RTX_Conf_CM.C
 *      Purpose: Configuration of CMSIS RTX Kernel for Cortex-M
 *      Rev.:    V4.70
 *----------------------------------------------------------------------------
 *
 * 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       0
#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

// </e>

// <e>User Timers
// ==============
//   <i> Enables user Timers
#ifndef OS_TIMERS
 #define OS_TIMERS      1
#endif

//   <o>Timer Thread Priority
//                        <1=> Low
//     <2=> Below Normal  <3=> Normal  <4=> Above Normal
//                        <5=> High
//                        <6=> Realtime (highest)
//   <i> Defines priority for Timer Thread
//   <i> Default: High
#ifndef OS_TIMERPRIO
 #define OS_TIMERPRIO   5
#endif

//   <o>Timer Thread stack size [bytes] <64-4096:8><#/4>
//   <i> Defines stack size for Timer thread.
//   <i> Default: 200
#ifndef OS_TIMERSTKSZ
 #define OS_TIMERSTKSZ  50
#endif

//   <o>Timer Callback Queue size <1-32>
//   <i> Number of concurrent active timer callback functions.
//   <i> Default: 4
#ifndef OS_TIMERCBQS
 #define OS_TIMERCBQS   4
```

```
#endif

// </e>

//   <o>ISR FIFO Queue size<4=>   4 entries  <8=>   8 entries
//                          <12=> 12 entries  <16=> 16 entries
//                          <24=> 24 entries  <32=> 32 entries
//                          <48=> 48 entries  <64=> 64 entries
//                          <96=> 96 entries
//   <i> ISR functions store requests to this buffer,
//   <i> when they are called from the interrupt handler.
//   <i> Default: 16 entries
#ifndef OS_FIFOSZ
 #define OS_FIFOSZ      16
#endif

// </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 (;;);
}


/*----------------------------------------------------------------------------
 *      RTX Configuration Functions
 *---------------------------------------------------------------------------*/

#include "RTX_CM_lib.h"

/*----------------------------------------------------------------------------
 * end of file
 *---------------------------------------------------------------------------*/
```

Listing 4: RTX_Conf_CM.c



Register	Value
Core	
R0	0x00000000
R1	0x10000098
R2	0x00000001
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x100002F8
R14 (LR)	0x00000000
R15 (PC)	0x0000092A
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Unprivileged
Stack	PSP
States	59327335
Sec	0.59360332

Show: Modules

Module/Function	Calls	Time(Sec)	Time(%)
part2		593.312 ms	100%

Option	Value
Thread Configuration	
RTX Kernel Timer Tick Configuration	
System Configuration	
Round-Robin Thread switching	

Round-Robin Thread switching
Enables Round-Robin Thread switching.

```
Load "S:\\COE718\\Lab_3a\\Part2\\part2.axf"
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALuate EXIT FUNC Go INCLUDE IRLOG ITMLOG KILL LogicAnalyze LOAD LOG MAP MODE Ostep PerformanceAnalyze Pstep RESET SAVE SCOPE SET
```

t1: 0.59360332 sec

Figure 2: Task Timing and Performance Analyzer