RTOS, Spring 2015 – Lab #7: Rate Monotonic

Paolo Torroni, paolo.torroni@unibo.it
Davide Chiaravalli, davide.chiaravalli@studio.unibo.it

Objective: to learn how to schedule hard tasks with fixed priorities.

1. Background

This lab builds on what we have learnt in Lab #6 about RTAI timers and task activations. Make sure you have understood concepts of periodic task scheduling with fixed priorities, in particular, the Liu-Layland and Hyperbolic bounds, and rate monotonic priority ordering.

2. Fixed priority scheduling

A) Define a taskset composed of N_TASKS = 5 tasks, in terms of phase Phi, period T, worst case computation time C, and priority P. These could be static global variables. You can use

http://lia.deis.unibo.it/Courses/RTOS/src/2015-7/ rmpo.c

as a template. There, you only need to set values.

B) Define a guarantee test based on Liu-Layland. If the task set is guaranteed, execute tasks for a hyperperiod. If it is not, visualize an error message. For simplicity, you can use the value in Ulub[5] for the least upper bound for 5 tasks.

C) Define a guarantee test based on the Hyperbolic Bound.

D) Modify the parameters of the taskset (for example: modify the periods or computation times), to obtain guaranteed/non-guaranteed tasksets.

E) Once you have a taskset that passes the guarantee test, apply RM to set priorities. Then execute the tasks.

Notes:

- You can use **rt_busy_sleep(RTIME n)** to simulate execution for a given number **n** of nanoseconds.
- To implement RMPO, you could sort the tasks based on their period. One possibility is to create a copy T1 of the values of the task periods T, then sort T1, and finally use the following code to set the priorities P of the taskset:

```
// assign priorities based on T1
for(i=0;i<N_TASKS;i++) {
    for(j=0;j<N_TASKS;j++) {
        if(T[i]==T1[j]) {
            P[i]=TOP_PRIORITY+j;
            break;
        }
    }
}
```

- For sorting a small array T1 of integer values, you can use the following code:

```
// sort T1
for(i=0;i<N_TASKS;i++) {
    min = i;
    for(j=i+1;j<N_TASKS;j++) {
        if(T1[j]<T1[min])
            min = j;
    }
    if(min!=i) {
        temp = T1[i];
        T1[i] = T1[min];
        T1[min] = temp;
    }
}
```

- Use the Kbuild file at http://lia.deis.unibo.it/Courses/RTOS/src/2015-7/Kbuild, which is already configured for rmpo.c
- Use rt_printk to display kernel log messages; however, notice that rt_printk does not handle floating point numbers (only integers or strings), therefore, if you want to display a floating point value, you can use **ftoa(double, int, char*)** to convert a fp number into a string and then use that string inside rt_printk. If the second argument is 0, ftoa shows 2 decimals; otherwise, it shows 6 decimals. The string variable must be defined before. For Example:

```
char str[40];
rt_printk("Computation time:%s ms\n", ftoa(1.234567, 0, str));
```

will add the following line to the kernel log:

```
Computation time: 1.23 ms
```

- Before you install your kernel module, be sure to install the RTAI modules:

  /usr/realtime/modules/rtai_hal.ko
  /usr/realtime/modules/rtai_sched.ko
  /usr/realtime/modules/rtai_math.ko

- If you want to display kernel log messages in the background, you can use

  sudo tail -f /var/log/kern.log &