Real Time Systems – 2008/2009

RTAI Tutorial

Preliminary Information for Assignment No. 1

Cristóvão Sousa
crisjss@gmail.com

v0.1 – September 30, 2008

1 Introduction

RTAI (RealTime Application Interface) is an extention for original Linux kernel which enables realtime services. This is made by means of RTAI tasks (RTASK), in kernel or user space. Kernel tasks run in kernel space within kernel modules. User space tasks are user applications threads which are switched to RTASKs with a given priority. Tasks can also be made periodic. The user space realtime framework is named LXRT.

Official documentation can be seen in RTAI Manual [1] which are not yet complete. The API can be seen in [2] or generated at RTAI compile time. Important LXRT functions of API are in:

- *LXRT module* in *Modules* section, and
- *An overview of RTAI schedulers*,
- *LXRT and hard real time in user space* and
- *LXRT-INFORMED FAQs* of *Related Pages* section.

Be carefull, some web documentation can be outdated. Also pay attention whether documented functions are to be used in kernel or user space.

2 Basic Tutorial

2.1 Applications Compilation and Execution

- Applications should be compiled with GCC [3–6]. The use of an automatic compile system, like Make [7, 8], is recommended.
- The main RTAI-LXRT header to be included is `rtai_lxrt.h`.

- LXRT programs need some special compile and linkage gcc flags which can be obtained, respectively, with:

```
/usr/realtime/bin/rtai-config --lxrt-cflags
/usr/realtime/bin/rtai-config --lxrt-ldflags
```

- When using POSIX threads, the flag `-D_REENTRANT` must be used for compilation and `-lpthread` flag for linkage.
- Before using realtime services, the RTAI kernel modules should be inserted. For basic user space functionality do:

```
sudo /sbin/insmod /usr/realtime/modules/rtai_hal.ko
sudo /sbin/insmod /usr/realtime/modules/rtai_lxrt.ko
```

- In order to run application in user space, root permissions are needed. To comply with this, the program can be executed by root (`sudo ...`), or the `rt_allow_nonroot_hrt()` should be called before any other in the code.

2.2 Principal Functions

- RTAI initialization functions are
  - `rt_set_oneshot_mode()`

  for mode selection, and
  - `start_rt_timer()`

  for scheduler start.
- A realtime task is a normal POSIX thread [9–11], which is switched to realtime scheduling with
  - `rt_task_init()` or
  - `rt_task_init_schmod()`

  commands, called inside the thread itself.
- Realtime mode switch is made with:
  - `rt_make_hard_real_time()` , and
  - `rt_make_soft_real_times()` .
- To turn a task in a periodic task use
  - `rt_task_make_periodic()` , or

  - `rt_task_make_periodic_relative_ns()` ,

  inside the task.

- The programmer is in charge of creating the periodic loop. To wait for each periodic activation use

  - `rt_task_wait_period()` .

  This function returns different values according to success of scheduling.

- Termination functions are `rt_task_delete()` and `stop_rt_timer()`.

2.3 Important Notes

- RTAI scheduler is activated by interruptions provided by an hardware timer. These interruptions can be periodic, which forces tasks to have periods multiple to scheduler period, or be automatically programed on demand. This can be controlled with `rt_set_periodic_mode` and `rt_set_oneshot_mode` functions before starting RTAI with `start_rt_timer()`.

- There are two time measure units in RTAI: nanoseconds and clock ticks. Clock ticks have duration dependent on scheduler mode. For conversions between this two measures `nano2count` and `nano2count` can be used [12].

- For time measures proper RTAI functions should be used. These functions names start with `rt_get_time_` and `rt_get_cpu_time_` [12], and have different resolutions.

- Functions `rt_task_make_periodic()` and `rt_task_make_periodic_relative_ns()` need a desired task beginning time as parameter. They differ in the form of this parameter. For `rt_task_make_periodic()` this time is in clock ticks relatively to RTAI scheduler start, i.e., in an absolute time measure (passing a past instant time will lead to errors). For `rt_task_make_periodic_relative_ns()` the parameter is in nanoseconds and is the desired interval of time between this function call and the first task activation.

- When running RTAI in multi processors machines, tasks can be spread through the various cores. To control which processor each task uses, a mask should be passed in function `rt_task_init_schmod()`. For the first assignment, the use of only one processor is mandatory.

References

[1] G. Racciu and P. Mantegazza, *RTAI 3.4 User Manual*, rev 0.3 ed., Oct. 2006. [Online]. Available: http://www.rtai.org

[2] (2005, Jan.) RTAI API documentation. [Online]. Available: https://www.rtai.org/documentation/magma/html/api/

[3] (2008, Sep.) GCC, the GNU Compiler Collection . [Online]. Available: http://gcc.gnu.org/

[4] V. Hegde. (2005, Nov.) Using the gnu compiler collection. [Online]. Available: http://linuxgazette.net/120/vinayak.html

[5] U. Clarkson. GCC quick reference. [Online]. Available: http://web2.clarkson.edu/class/cs241/labs/lab_1/using_gcc.html

[6] B. Gough. An introduction to GCC. [Online]. Available: http://www.network-theory.co.uk/docs/gccintro/index.html

[7] (2006, Apr.) GNU Make manual. [Online]. Available: http://www.gnu.org/software/make/manual/

[8] B. Yoshino. (2005, Feb.) Make - a tutorial. [Online]. Available: http://www.eng.hawaii.edu/Tutor/Make/

[9] D. Robbins. (2000, Jul.) Posix threads explained. [Online]. Available: http://www-128.ibm.com/developerworks/linux/library/l-posix1.html

[10] G. Ippolito. (2004) Posix thread (pthread) libraries. [Online]. Available: http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html

[11] (2008, Sep.) POSIX Threads. [Online]. Available: http://en.wikipedia.org/wiki/POSIX_Threads

[12] RTAI API – RTAI schedulers modules – Timer functions. [Online]. Available: http://download.gna.org/rtai/documentation/vesuvio/html/api/group___timer.html