Advanced RTAI Features
LXRT

LXRT Overview

- User-Space interface to the facilities and features of RTAI.
  - Provides symmetric API that may be used by both real-time RTAI tasks and Linux processes.
- Allows the user to create a real-time task using RTAI's API from user space.
  - Ability to recover after the crash of a Linux process with a real-time LXRT component.

LXRT Overiew

- Created real time task can be
  - Soft real time (run under Memory Protection of Linux)
  - Hard real time (run within the kernel having access to kernel memory)
- Allows dynamic switching of tasks between the hard/soft real-time modes from within the application
- Linux System calls are available for a soft real time task created in User Space

LXRT Overview

- ‘soft’ real-time performance in User Space is acheived by using
  - SCHED_FIFO scheduling policy and statically assigned process priorities from 1 to 99 for LXRT processes
  - SCHED_OTHER policy for other linux processes with static priority 0

LXRT Overview

- ‘soft’ real-time tasks can miss their deadlines when preempted by interrupts or real-time activities in kernel space.
- Switching to hard real time mode (scheduled by the RTAI scheduler)
  - Creates a real-time agent task on behalf of LXRT process. Using RTAI API
  - Execute real time services and communicate back

LXRT Overview

- Benefits
  - Provides protection against system crashes during the development and debugging phases
  - Tasks can be debugged using user-space debug tools.
  - Simpler to move tasks between the hard and soft real-time domains (due to the use of standard RTAI API)

LXRT API

Create a LXRT Real time Agent Task

```
RT_TASK *rt_task_init(
        int taskname,
        int priority,
        int stack_size,
        int max_msg_size );
```

- name is a unique identifier and can be created using the '*name2num*' macro, e.g.:

```
int taskname = nam2num("MTASK1");
```

LXRT API

start_rt_timer(RTIME period)

- Starts the timer for scheduling

rt_is_hard_timer_running()

- Checks if the timer is started

print_to_screen(const char *format, ...)

- Prints information from hard real-time user space task to the screen

LXRT API

void rt_make_hard_real_time(void)

- Translates a soft real-time Linux process into a hard real-time LXRT process

void rt_make_soft_real_time(void)

- Returns a hard real-time LXRT process to soft real-time Linux process.

int rt_task_delete(RT_TASK *task)

- Delete the real-time agent task

LXRT Process

- Create a 'soft' real time linux process

```
int main()
{
        struct sched_param mysched;
        mysched.sched_priority = 99;
        if (sched_setscheduler ( 0, SCHED_FIFO, &mysched )
            == -1 ) {
                puts(" Error in setting the Scheduler policy ");
                exit(1);
        }
```

LXRT Process

- Create Real time Agent task

```
RT_TASK *mtsk,
unsigned long mtsk_name =       nam2num("SRV");
if (!(mtsk = rt_task_init(mtsk_name, 0, 0, 0))) {
        printf("CANNOT INIT SRV TASK\n");
        exit(2);
}
```

LXRT Process

- Start timer and set task to be periodic

```
if ( rt_is_hard_timer_running() ) {
        start_rt_timer(nano2count(1E5));
}
rt_task_make_periodic(mtsk, rt_get_time(),
   nano2count(1E9));
```

- Make task hard real time

```
rt_make_hard_real_time();
```

LXRT Process

- Periodic Execution

```
while (keep_on_running)
{
        // Do task's activities
        rt_task_wait_period();
}
```

- Delete the task

```
rt_task_delete(mtsk);
```