

## Stand-Alone OCERA RTLinux

**Stand-Alone RTLinux (saRTL)** is a Linux independent implementation of RTLinux which provides the execution environment based on POSIX for critical applications with a minimal executive.

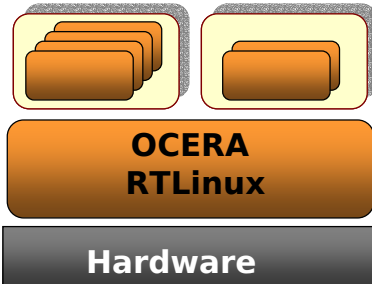
**saRTL** support 3 memory protection schemes:

- ❖ **Flat memory without paging.** This is the default option. This is the memory scheme used by RTLinux where executive memory is accessible from any thread and all threads are able to access the whole memory.
- ❖ **Executive memory protection** (optional functionality). This model protects the RTLinux executive against write access from application threads. Application threads are not protected among them.
- ❖ **Context memory protection** (optional functionality). This is the more flexible model. Several contexts are created, each context contains one or more threads. Each thread will have total access to all threads within its context but it will have only read access to other contexts and executive.

### Tools

- ❖ **GDB Debugging Agent:** It is a small piece of code running on the target that helps ***gdb*** to carry out requests to monitor and control the application being debugged. The implemented agent jointly with the GDB debugger opens the possibility to use all the power of the GDB debugger in **saRTL** applications. Among other abilities to step by step execution of thread code and executive code; insert breakpoints; display and modify the variable values; etc. This facility is available even when the interrupts are disabled.
- ❖ **saRTL Tracer:** A non-POSIX compatible tracing utility which introduces low overhead to the embedded system.
- ❖ **RT-Terminal:** It enables RTLinux applications to display data directly on the console screen and to read the value of a pressed key directly from the keyboard,

Hard Real-Time Tasks running in different memory protected spaces



### Features

- ❖ **Low memory overhead.** The core system contains only the following modules: the code of RTLinux, the minimum code needed to boot the system and basic virtual memory management.
- ❖ **Scalable.** Users will be able to customize RTLinux functionalities to reduce kernel memory usage.
- ❖ **Porting will be possible to systems without hardware for virtual memory support.** Therefore, Stand-Alone RTLinux could be ported to a wider range of architectures.
- ❖ **Less TLB and memory cache misses** since only real-time applications are being executed.
- ❖ **One problem of RTLinux is the use of kernel modules or applications that execute directly interrupt management processor instructions** (*cli* and *sti*), for example, some *xfree86* drivers disable interrupts or lock the PCI bus for long time periods. Stand-Alone RTLinux removes this problem since the target system will be compiled to a single static and bootable kernel.
- ❖ **It is not longer limited by the Linux memory manager.** So, it is possible to implement custom virtual memory management algorithms. Stand-Alone RTLinux has full control of the MMU.