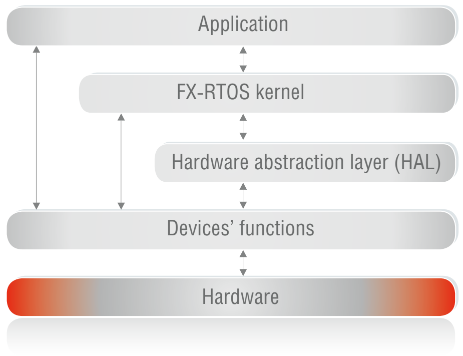
FX-RTOS

FX-RTOS — real-time operating system for embedded applications

- Component architecture without mandatory elements. Unused components are not included into the target image
- Symmetric multiprocessing is supported
- The system scales down to simple preemptive event handlers without full-fledged threads
- The operating system is configurable using dependency injection technique
- Reusable components can be combined in various configurations
- Various scheduling policies are supported (priority-based, FIFO, round-robin and their combinations)
- Extensibility with user-defined components
- Effective interrupt handling
- Easy cross-platform portability
- POSIX support (optional)
- Cross-platform utilities for configuring and building the OS
- Source code is provided



Component architecture

FX-RTOS is not a monolithic program. It defines a set of loosely coupled components for building your own OS. Configuration utility analyses components dependencies and builds an image (library) that exactly matches application needs without any extra overhead.

Multiple implementations of a single component interface are allowed. An appropriate implementation may be chosen during configuration to fit application system requirements. Some of the components may be used separately, without the OS itself.

Configuration process relays on special dependency injection technique. A component source code is not affected by configuring the OS.

Configuring is performed at compile time, so resulting machine code is almost optimal.

Multiprocessor support

FX-RTOS was initially designed with multiprocessing in mind. Thus, the amount of global variables shared between processors is reduced to minimum, there are no global system-wide locks. Therefore, the system performance scales well with number of processors.

For multiprocessor case specially designed components are used, so it does not affect performance of uniprocessor systems.

Special support for simple event driven systems

Many embedded systems are event-driven and may be represented as a set of event handlers. A handler, if already running, never waits. Such systems may be implemented with threads, but threads are too heavyweight and have redundant functionality that isn't used in this simple case.

FX-RTOS provides a special profile for such systems. Threads are removed and replaced with event service routines (ESR) similar in some aspects to program interrupts. Using ESR instead of threads reduces code size by 40%, increases performance and reduces RAM usage (in contrast with threads, ESR may share a single stack).

Extensibility

As several implementations of a single component interface are allowed, the kernel functionality may be extended or modified with components, developed by OS user or community.

Effective interrupt handling

FX-RTOS supports several techniques facilitating interrupt handling:

1) interrupt service routines (ISR);
2) deferred procedure calls (DPC);
3) threads (interrupt service threads, IST);
4) event service routines (ESR).

By configuring the OS, the system designer may decide to include an appropriate combination of these facilities into the target system depending on preferred application design principles and requirements. Interrupt handlers (ISR) run on special interrupt stack (this reduces stack size should be reserved for threads).

.

EREMEX
Innovative Software Solutions