

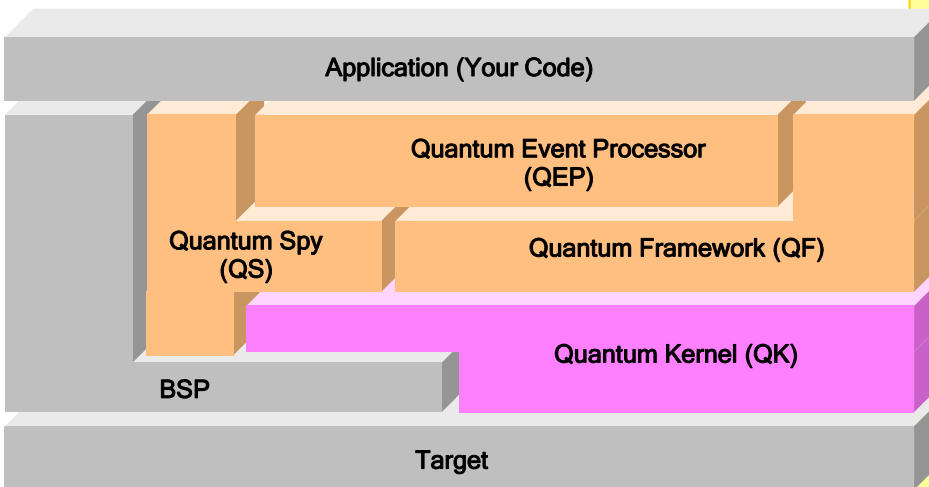


Quantum Kernel™ (QK) is a tiny preemptive, priority-based real-time kernel specifically designed to execute state machines in run-to-completion (RTC) fashion.

Benefits

QK ensures that the highest-priority task ready-to-run is always given control of the CPU. In other words, without the preemptive kernel the CPU may not execute a task that is time critical because it would be working on something that is not.

The biggest advantage of the preemptive **QK** is **deterministic** processing of time critical events, which is almost insensitive to software changes in lower-priority tasks.



Simplicity and Efficiency

Unlike other preemptive kernels, **QK** does not require any specific stack frame for interrupts. This means that you can write ISRs in C, without the need for assembly programming. The following code snippet shows the ISR for the ARM processor:

```

__irq __arm void tickIRQ(void) {
    uint8_t pin; // temporary for the QK priority
    T1IR = 0x1; // clear the interrupt source
    QK_IRQ_ENTRY(pin, TICK_IRQ_PRIORITY);

    QF::tick(); // process the QF clock tick

    QK_IRQ_EXIT(pin, (VICVectAddr = 0));
}

```

Features

- Preemptive, priority-based, **deterministic** execution of up to 64 tasks;
- Run-to-completion event processing without possibility for blocking;
- Use of **single stack** for all tasks and interrupts on most processor architectures;
- Priority-ceiling mutex support;
- Independence on exact stack frame layout for interrupts, which results in very C-friendly interrupt handling.
- No need for assembly coding of ISRs for most embedded compilers capable of generating interrupt service routines (ISRs) in C;
- Smaller and faster context switch than most traditional preemptive kernels;
- Full integration with Quantum Framework® (QF);
- Extremely small memory footprint around 150 bytes of ROM for the core scheduler and just 10 bytes of RAM for the task-ready-list and current priority;
- Easily portable to any CPU and compiler;
- Very clean, lint-free source code 99%-compliant with the Motor Industry Software Reliability Association (MISRA) guidelines;
- Dual-licensing model with choice between the **open source** license (GPL) and flexible, royalty-free commercial licensing options.

