Electronic Design

LEE GOLDBERG, Contributing Editor

Which Embedded RTOS is Right for Your Application?

Whether it's the safe operation of a motor vehicle or safe arrival of a spacecraft on a distant planet, there's an RTOS that can make sure it gets there. Here's a field guide to real-time software that can help you identify the architecture and features best suited to your project.

In the beginning, real-time operating systems (RTOSs) were primarily used in military, aerospace, and high-end industrial control applications. This has changed dramatically as low-cost, fast, and abundant computing power made embedded real-time computing practical for a growing number of traditional and IoT applications.[1]

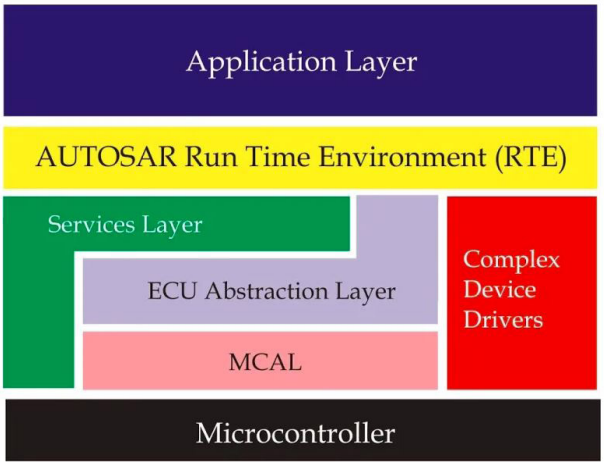
In recent years, RTOSs have become easier to use and available with large libraries of development tools and application-specific stacks. Likewise, the RTOSs themselves have undergone differentiation, with architectures and features that are optimized to suit a specific type of application or platform *(Fig. 1)*.

To help you find an RTOS that's a good match for your application, we've curated a small sampling of RTOSs that cut across the broad application space where real-time embedded computing is commonly used today.

AUTOSAR

If your real-time application involves any type of vehicle, AUTOSAR should be on your short list of candidate RTOSs. First released in 2003, AUTOSAR (AUTomotive Open System ARchitecture) was developed as an open platform that

1. Whether your application is responsible for the safe operation of a motor vehicle or the safe arrival of a spacecraft on a distant planet, there's an RTOS that can make sure it gets there. (Credit: NASA Jet Propulsion Labs)



2. Shown is a block diagram of the AUTOSAR OSI. (Credit: AUTOSAR)

3. The INTEGRITY architecture supports multiple protected virtual address spaces, each of which can contain multiple application tasks. (Credit: Green Hills Software)

addressed some of the emerging issues stemming from the rapid adoption of embedded computing in mission-critical automotive systems.

With a typical vehicle containing several dozen (or more) embedded processors, many of which must work closely together, the industry needed a common, hardware-independent programming platform that supported real-time functionality and met the most stringent standards for safety and reliability.

To address these requirements, AUTOSAR was based on a layered architecture similar to an OSI model, with separate layers of code that handle and abstract different types of operations that communicate with each other via standard APIs *(Fig. 2)*.

Since AUTOSAR is primarily used in automotive applications, it's been designed from the ground up to make efficient use of the limited memory and resource-constrained MCUs found in most automotive electronic control units (ECUs). It also provides native support for commonly used automotive buses, including CANbus, FlexRay, and Ethernet.

AUTOSAR's layered architecture includes a layer dedicated to support hardware functionalities called **MCAL** (microcontroller abstraction layer), which has drivers for accessing the underlying hardware peripherals of an MCU. As a result, most application code can be written independent of the hardware, allowing it to be run on different hardware platforms. It also provides a formal standard for communication between ECUs, enabling them to communicate with each other irrespective of the ECU's developer or the MCU it's based on.

To address the challenges involved with advanced safety features, driver assist, and automated driving functionality, AUTOSAR also offers the AUTOSAR Adaptive Platform. It's designed to meet the requirements of highly automated vehicles and supports dynamic updates and reconfigurations of software systems.

FreeRTOS

FreeRTOS is a very popular real-time operating-system kernel for microcontrollers and small microprocessors used in embedded devices, with an emphasis on reliability and ease of use. It's usually written in the C programming language, only requiring a minimum of assembly language instructions, mostly for its architecture-specific scheduler routines. This makes the kernel easy to port and maintain across multiple processor architectures.

At present, FreeRTOS is supported by 15+ toolchains and over 40 MCU architectures, including the latest RISC-V and ARMv8-M (Arm Cortex-M33) MCUs, such as Texas Instruments' Sitara series.[2]

FreeRTOS implements multiple threads by having the host program call a thread tick method at regular short intervals. The thread tick method switches tasks depending on priority and a round-robin scheduling scheme. The usual interval is 1 to 10 ms (1/1000 to 1/100 of a second) via an interrupt from a hardware timer, but this interval can be changed to suit a given application.

Distributed freely under the MIT open-source license, FreeRTOS includes a kernel and a growing set of libraries that cater to the specific requirements of IoT applications. The OS, and its libraries of drivers, security updates, and application-specific add-ons, are maintained by Amazon Web Services (AWS) for the benefit of the FreeRTOS community.

As part of its stewardship activities, Amazon provides an extension of FreeRTOS, referred to as *a:FreeRTOS*. It includes libraries that support IoT functionality, specifically for Amazon Web Services.

Two other licensed variants of FreeRTOS are also available:

4. In addition to helping automate the Air Force's drones and in-air refueling systems, VxWorks has a long history of use in space-bound hardware, including the Mars Insight Lander and the Mars Curiosity Rover. (Credit: VxWorks)

- OpenRTOS is a commercially licensed version of the FreeRTOS kernel that includes indemnification and dedicated support. FreeRTOS and OpenRTOS share the same code base.

SAFERTOS is a derivative version of the FreeRTOS kernel that's been analyzed, documented, and tested to meet the stringent requirements of industrial (IEC 61508 SIL 3), medical (IEC 62304 and FDA 510(K)), automotive (ISO 26262), and other international safety standards. SafeRTOS includes independently audited safety lifecycle documentation artifacts.

INTEGRITY RTOS

Created and maintained by Green Hills Software, the INTEGRITY RTOS is one of the oldest (30+ years), most feature-rich, and battle-hardened embedded RTOS platforms on the market *(Fig. 3)*. With a long heritage of use in aerospace, military, and other big-ticket, mission-critical applications, it's no surprise that INTEGRITY and its related add-ons are on the high end of the RTOS cost continuum. But, in many cases, the added security and functionality it offers are well worth it.

For one thing, Integrity's architecture uses hardware memory partitioning that prevents processes from writing beyond assigned memory regions. This helps ensure that the RTOS kernel is secured against interference from application errors as well as malicious code—including denial-of-service attacks, worms, and Trojan horses. Likewise, each application is run in its own block of protected virtual memory space, limiting its exposure to malicious tampering and its ability to cause problems with other system elements.

To maintain top performance, INTEGRITY employs a real-time scheduler that supports multiple priority levels, enabling positive control over CPU allocation. To make the most efficient use of the system's processing resources, its kernel services routines are optimized to streamline system calls, enabling them to be quickly suspended to allow for the execution of other higher-priority calls.

Another example of INTEGRITY's reliability is that the RTOS kernel has a built-in, isolated memory stack to eliminate the possibility of corruption due to user stack overflow.

In addition to a deep library of middleware that supports everything from multiple filing systems to encryption and web services (HTTPS, SOAP, AJAX, JSON, XML), INTEGRITY provides advanced support for using multicore processors to implement applications using asymmetrical multiprocessing (AMP) and symmetrical multiprocessing (SMP). Should your application require it, INTEGRITY also offers optional support for a multivisor system.

And, if your project requires an even higher level of security, consider opting for Green Hills' INTEGRITY-178/178 tuMP. They're both variants of the original INTEGRITY RTOS that support multiprocessing architectures in high-security, mission-critical mil/aero applications. They've been certified to the highest levels of airborne safety (DO-178B/C DAL A) and security (SKKP/EAL 6+) for more than 80 airborne systems.

VxWorks

First released in 1987, Wind River's VxWorks is arguably the earliest player in the RTOS market. Originally developed for mil/aero applications, it offers board support packages (BSPs) and software stacks that comply with virtually all critical industry certification standards, such as DO-178C, IEC 61508, IEC 62304, and ISO 26262 *(Fig. 4)*.

The deterministic, priority-based preemptive RTOS was designed to deliver high reliability with low latency and minimal jitter. It provides efficient, deterministic execution using a combination of scheduling mechanisms that include

- Priority-based preemption with optional round-robin
- Time and space partitioning
- Adaptive scheduling offering foreground and back-

5. The RTOS Zephyr RTOS provides a lightweight, reliable platform for the firmware that drives the TZero IoT module, a simple way to collect data from industrial processes. (Image: Zephyr Foundation)

ground threading
- OSIX PSE52 thread scheduling extensions (FIFO and sporadic)

VxWorks supports most MCUs based on the AMD/Intel architecture, POWER architecture, Arm architectures, and RISC-V. It can also be used in applications requiring either multicore asymmetric multiprocessing (AMP) or symmetric multiprocessing (SMP), as well as mixed modes. It supports support multi-OS (via a Type 1 hypervisor) designs on 32- and 64-bit processors.

In addition to hardware-partitioned kernel and user space environments, VxWorks bristles with many other architectural features that enhance reliability and security that include:

- Secure boot (digitally signed image)
- Secure ELF loader (digitally signed applications)
- Kernel hardening (non-executable pages, stack guard pages, stack smashing protection, etc.)
- Secure storage (encrypted container + full disk encryption)
- Address sanitizer (ASan)

The latest release of VxWorks includes support for most modern development languages, including C++17, Boost, Rust, Python, pandas, and more. It also includes an edge-optimized, OCI-compliant container engine that's compatible with the industry's initiative to standardize container formats and runtimes.

Zephyr

Zephyr is an open-source RTOS created and supported by the Linux Foundation's Zephyr Project. It's often used in small, lightweight platforms like the TZero IoT module *(Fig. 5)*.

This versatile RTOS is true to its LINUX roots, providing a truly hardware-independent platform that's easy and efficient to deploy, secure, connect, and manage. It is built with an emphasis on broad chipset support, security, dependability, long-term support releases, and a growing open-source ecosystem. Unless your application has some very demanding, application-specific requirements, this RTOS is very worthwhile to consider.

Other Options

While this diverse sampling represents only a small fraction of the RTOSs available on the market, it should give you a good first-order understanding of the types of solutions available, and the types of applications they can be used in. Here are a couple of others that are also worth exploring:

- Microsoft's Azure is created specifically for deeply embedded applications with the goal of providing reliable, ultra-fast performance for resource-constrained devices. It offers real-time multithreading, inter-thread communication and synchronization, and memory management.
- Deos is a safety-critical time and space partitioned RTOS developed by DDC-I specifically for avionics and other advanced aerospace applications. It's verified to meet the guidance of DO-178C/ED-12C Design Assurance Level A (DAL A) for Avionics Applications as well as support the ARINC 653 APEX and Rate Monotonic Scheduling (RMS) standards.

References

1. "3 trends impacting the future of embedded processing technology," Texas Instruments.
2. "Automotive quad-core Arm Cortex-R5F MCU up to 400 MHz with real-time control and security," Texas Instruments.