print | close

Integrating POSIX and ARINC in FACE-Aligned Operating Systems

Electronic Design

Greg Rose

Mon, 2016-11-14 16:19

Download this article in .PDF format
This file type includes high-resolution graphics and schematics when applicable.

The FACE standards effort seeks to facilitate rapid deployment and lower cost by moving military/aerospace systems development from a “stove-pipe” approach to a set of interoperable and reusable avionics components. Instead of redesigning modules and software for each new system and proprietary set of APIs, the prime contractor can now leverage a set of FACE software building blocks.

The heart of any safety-critical avionics system is the operating system on which it’s hosted. As part of the FACE initiative, the consortium developed a base profile for RTOSs that combines ARINC 653 and POSIX. FACE-aligned operating systems (Security, Safety Base, and Safety Extended) are expected to provide hard partitioning between software subsystems as well as ARINC 653 APIs and a subset of POSIX APIs

The Security profile, which sets the baseline for the FACE Technical Standard’s Operating System Segment (OSS), includes the ARINC 653 API and has the lowest level of POSIX functionality. The Safety Base profile adds an expanded set of POSIX APIs. The Safety Extended profile continues with the ARINC 653 as in the other profiles and adds multi-process support with an even richer set of POSIX API.

Related

The Military FACEs Its High-Tech Future

Memory Partitioning and Slack Scheduling Boost Performance in Safety-Critical Applications

The RTOS Motto: On Time And On Budget

These three profiles target systems typically expected to be used in mission-critical safety and/or security-conscious applications with certification requirement. They require a combination of the features historically found in certified ARINC 653 operating systems, as well as the POSIX APIs historically found in real-time operating systems.

The FACE OSS is unique in that it requires the RTOS platform to support both the ARINC 653 and POSIX standards. The ARINC 653 interface provides the rigid fixed-in-time scheduling required for tasks with high safety criticality, such as avionics applications. The POSIX interfaces enable developers to quickly access third-party code for less-critical event-driven functions, such as maintenance and networking, with possibly lower safety criticality requirements in the system.

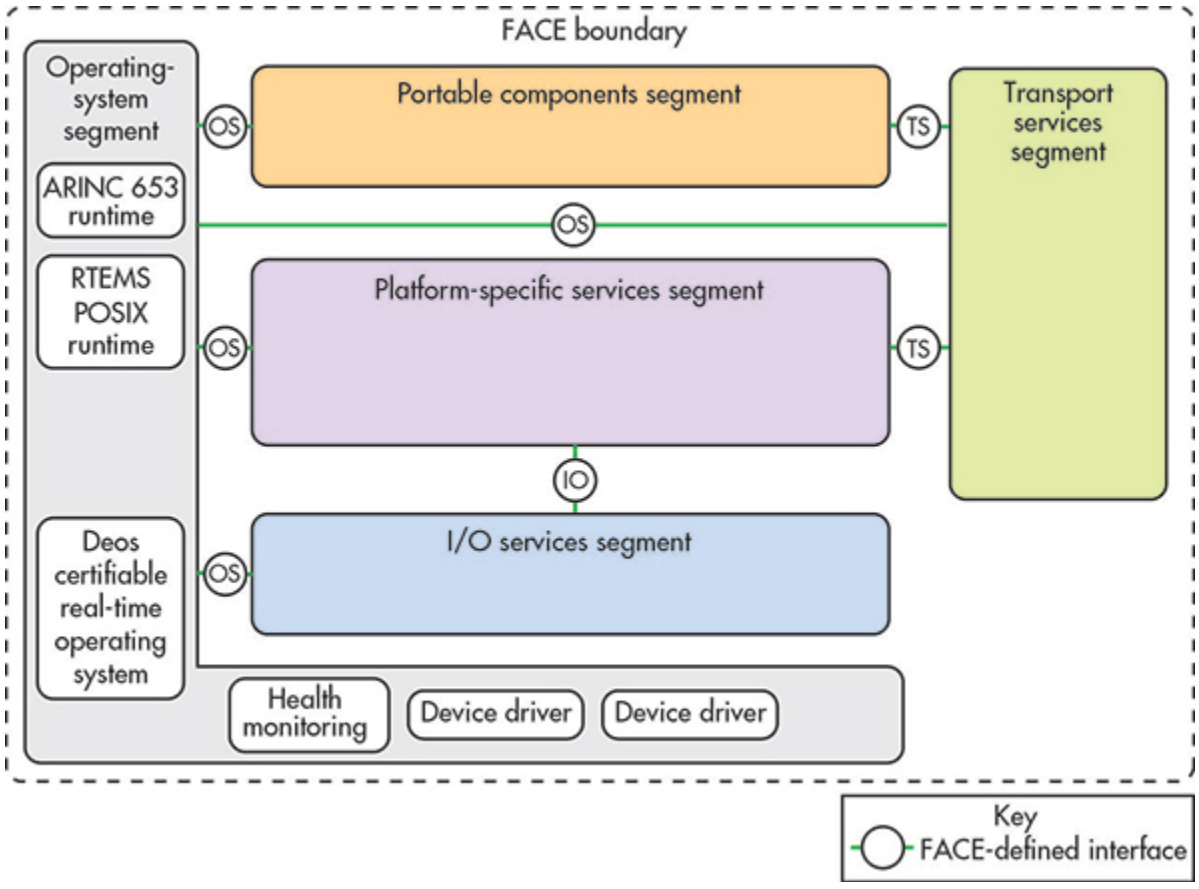
veloping a single platform that supports both POSIX and ARINC 653 is a challenge because the two APIs e typically been targeted and optimized for different application spaces. One way to address both APIs is to use a DO-178-certifiable, safety-critical partitioned RTOS and allow POSIX run-time environments to run within the partitions along with ARINC 653 within other partitions. DDC-I and On-Line Applications Research (OAR) have taken this approach, providing an integrated solution for the FACE Safety Base Profile that incorporates ARINC 653 and POSIX functionality running on DDC-I's Deos.

The integration features OAR's RTEMS hosted in a Deos time partition, giving safety-critical developers a DO-178C-certifiable RTOS solution that delivers hard real-time response, time and space partitioning, and both POSIX and ARINC-653 interfaces. In this configuration, Deos provides memory partitioning and ARINC 653 scheduling. RTEMS supplies the Safety Base POSIX APIs, immediately fulfilling the vast majority of the API requirements for the FACE OSS Safety Base profile.

RTOS Responsibilities

Hard real-time systems require that a set of tasks, or threads, interleave with predictable execution time bounds, known as deadlines. For hard real-time tasks, every deadline must be met, while soft real-time tasks might miss deadlines or put bounds on the maximum tardiness of meeting the deadline.

Often, the set of tasks contain a mix of periodic and aperiodic tasks. Periodic tasks have a period that controls the rate at which they're released to execute, each task having a defined time budget that's equal to its worst-case execution time (WCET). All other tasks are aperiodic.



The primary function of a hard real-time operating system (RTOS) is to help developers of embedded-system applications ensure that schedulability guarantees are met for hard real-time safety critical tasks. A secondary goal is to achieve high quality of service for soft real-time and other aperiodic tasks. Other common RTOS

ctions provide services required for application execution, coordination, and communication, such as chronization, memory management, file systems, IO devices, networking, and more.

Hard Partitioning

A RTOS must be both reliable and robust. We expect software designed to the guidance of DO-178 Design Assurance Level A to be reliable through correctness of the software. However, the RTOS must also be able to withstand application failures from lower Design Assurance Level (DAL) software running on the same device. The failure of an application can't corrupt the execution of the kernel or its ability to provide services to other applications. Kernel robustness is required and verified by rigorous software test, where the RTOS must correctly deliver services despite the varied failure modes that are possible from the applications software.

The RTOS accomplishes this by building walls around applications contained in a process/partition. An application within a process/partition cannot break out of its walls to steal another application's processing resources or otherwise interfere with its execution. The strength of these walls allows the RTOS to concurrently execute software at different DO-178C levels, A through E.

Systems that were historically "control only" now can include maintenance and monitoring functions safely integrated. Historically, basic maintenance and monitoring functions were run on a PC-style device run on the ground. But now with the advent of higher-performance processors, safely partitioned operating systems, and industry-standard interfaces, more advanced control, maintenance and monitoring functions can run on a single device in flight.

Imagine being able to continuously monitor more data in real time and flag potential wear indicators or system concerns early, before they become larger more costly problems to solve, or even catastrophic failures. With the addition of POSIX interfaces, existing lower design assurance code can be easily ported to the device and partitioned off in a low DAL (not flight-critical protected) partition to perform monitoring.

RTOSs such as Deos afford an extra level of protection that prevents developers of lower design assurance code or less-experienced developers from allocating and accessing critical portions of memory. Typically, it's the least-skilled developer who poses the greatest risk to system integrity.

The Deos platform integrator prevents novice developers who would likely be working on tasks with the lowest safety-criticality requirements from allocating memory reserved to critical tasks or critical resources. It does so by defining their access privileges in the system registry.

In addition to providing scheduling, the RTOS manages computing resources such as processing time, physical memory, I/O, and interrupts. It also manages kernel resources like processes, threads, semaphores, mutexes, events, and mailboxes. Deos guarantees that all resources allocated in the system registry are available to the process/partition using the resources. The Deos integration tool used at build time ensures that system resources aren't over-allocated.

Bridging ARINC 653 and POSIX for FACE Alignment

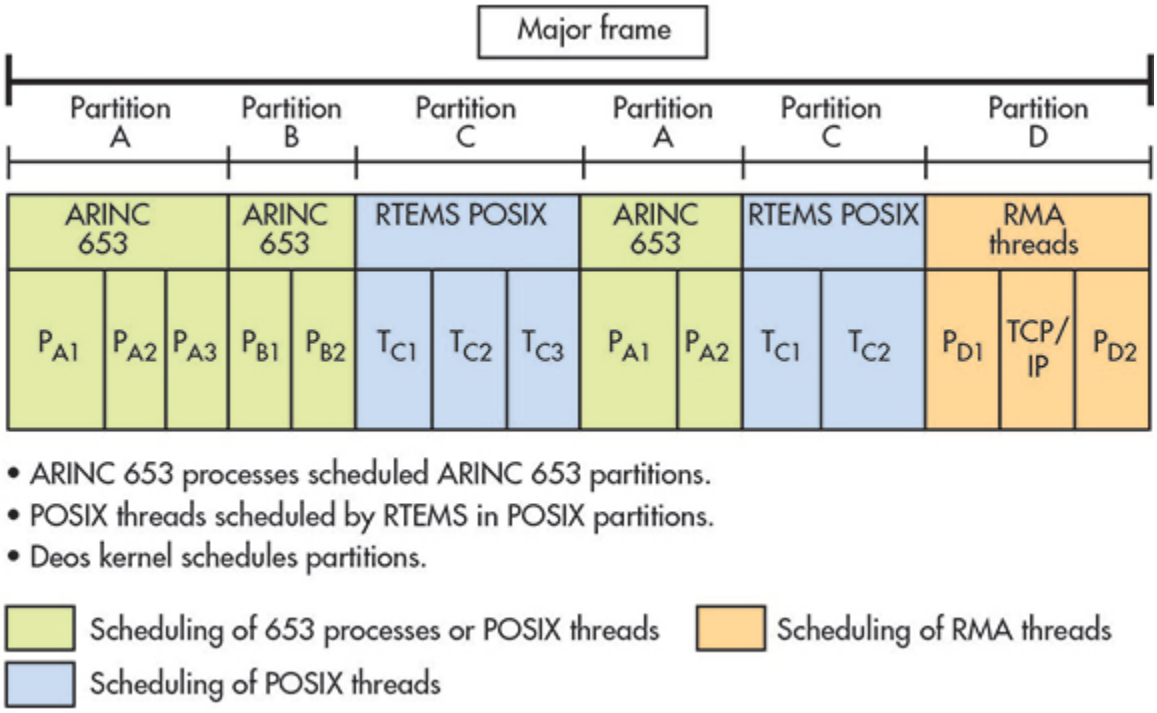
Standardized APIs that define OS functions and parameter passing have existed for years (such as POSIX and ARINC 653). FACE takes that one step further by also defining the communications interfaces (as a part of the Transport Services Segment (TSS)), and the Data Model etc. The goal is to make the software interoperable by Units of Portability (UoP), and enable both legacy and new software, whether written to POSIX or ARINC 653, to coexist on the same system.

Figure 1 shows a combined Deos/RTEMS FACE solution. Here, Deos provides ARINC 653 time, space

titioning, and TCP/IP services for user applications that meet the FACE Safety Base Profile requirements.

RTEMS, a native POSIX 52-compliant RTOS (single process/multiple threads), serves as a classic hard-real-time executive, providing POSIX API services that facilitate application portability while maintaining a deterministic real-time environment. RTEMS also provides non-POSIX RTEMS services such as stack checking, performance monitoring, a shell, and the Classic API based on the RTEID and ORKID specifications. Many of these services are helpful while debugging and tuning a system, but should not be included in production builds by applications that must be FACE-aligned.

The integration of Deos and RTEMS is achieved through the RTEMS/Deos BSP and Adapter. The BSP provides the hardware abstraction layer that tailors RTEMS for a specific target board. In this case, the target board is the virtualized environment provided by the Deos kernel.



RTEMS and applications hosted on RTEMS execute in a protected address space. This requires that the RTEMS be built in paravirtualized mode, which allows it to execute without the use of privileged instructions. The BSP initializes the stack and registers, providing a logical abstraction that uses Deos kernel services to implement services that would normally be done via direct hardware access.

With proper adaptation and sufficient device abstraction, all RTEMS services can be supported in this environment. However, the primary focus is on multithreading, including communication and synchronization services.

The Deos kernel provides the overarching time, space, and resource partitioning required to keep the applications in each partition safe from anything running in other partitions in the system. The scheduling of the partitions follows the ARINC 653P1 specification as shown in *Fig. 2*. Partitions are scheduled in a timeline per major frame. Within each partition, the units of execution are scheduled via a second level scheduler such as ARINC 653, RTEMS, or, in the case of the TCP/IP stack, Deos' native RMA scheduler.

Portability, Reuse, Upgradeability

establishing standards for software interfaces, interoperability, and certification, FACE will reduce vendor
ς, opening what have historically been sole-sourced software solutions from one vendor to interoperable solutions from multiple suppliers. This increased competition not only lowers per-program cost, but also makes it easier for program managers to take advantage of best-in-class technology and services.

The new standards will also enhance portability and reuse. This further reduces cost by making it easier to utilize software components across multiple platforms and programs.

The beauty of the Deos/RTEMS integration is that the Level A certifiability of Deos remains unchanged, as does the open-source nature of RTEMS. In fact, each RTOS can be used in its existing form when desired. As a result, there's no impact on existing applications for either RTOS.

RTEMS can still be used to host applications on bare metal. It's also able to host FACE-aligned applications (with restrictions) that can be easily migrated to the fully FACE-aligned Deos/RTEMS environment.

Source URL: http://electronicdesign.com/dev-tools/integrating-posix-and-arinc-face-aligned-operating-systems