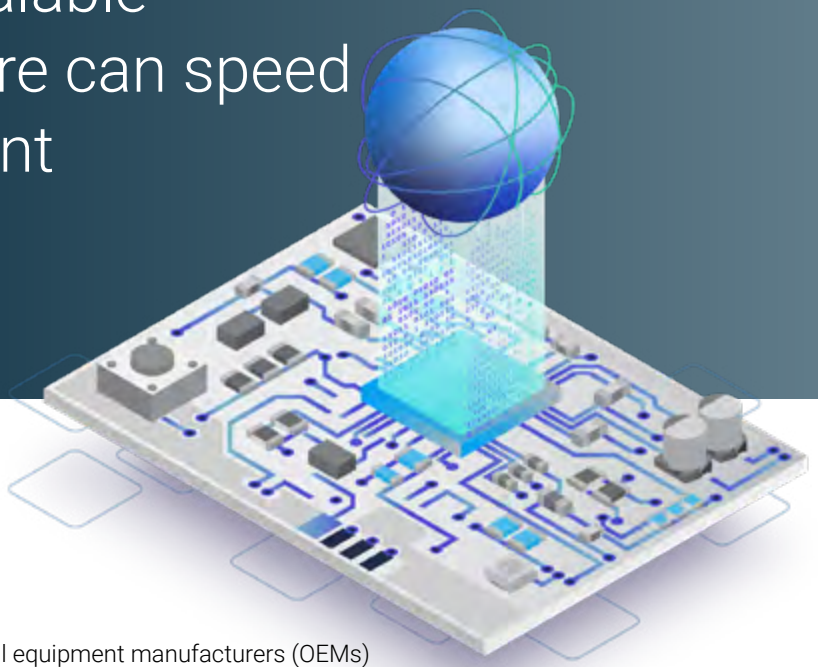
BlackBerry | QNX

Accelerate Innovation:
How adopting a scalable platform architecture can speed product development



The pace of innovation has changed. Original equipment manufacturers (OEMs) can no longer take years to introduce new products or update functionality in existing products. If your customers can't get what they want, when they want it, they're quick to go elsewhere.

OEMs in nearly every industry are becoming software-driven. But while software can provide differentiation, inefficiencies in internal development teams can throttle product innovation and lengthen time to market. These inefficiencies are often based on a growing array of one-off design decisions, each leading to different hardware and software technology choices that need to be managed and supported by some of the company's most valuable development resources. OEMs need their software development organizations to quickly and cost-effectively improve their competitive position through new product introductions (NPIs) and by adding in-demand features and capabilities to existing product lines.

One solution is the migration to a standard, scalable platform based on a single processor architecture and commercial operating system (OS). With a standard platform architecture for all products, OEMs can take advantage of proven methods to accelerate product innovation and time to market, make better use of limited developer resources, and reduce development inefficiencies and costs.

Consolidation, Connectivity, and Complexity are Driving Change

While many OEMs have attempted to make incremental shifts in their development approach, the convergence of three technology trends has increased the urgency for a foundational change.

01/

Consolidation drives performance, cost and efficiency. The microprocessor is typically one of the earliest design decisions for development teams, who often make these decisions independently of other programs, resulting in little commonality across product lines. But today's powerful and cost-effective processors allow multiple functions to be consolidated on a single board or system. In response, OEMs are standardizing on a single processor architecture, such as Arm® or Intel® x86, which offer a range of cost and performance levels, with common development environments and extensive ecosystems.

02/

Connectivity supports new functionality and revenue streams but adds security concerns. Today's consumers expect internet connectivity in almost any product, often with cloud-based services and applications to add functionality. This expectation aligns with OEMs' need for analytics to better understand consumer needs and to support new revenue streams from add-on services. Connectivity also requires new software, such as analytics, dashboards and human machine interfaces, and increases security risks, making it critical to identify and mitigate software vulnerabilities before a product is released.

03/

Complexity demands more of already-stressed development resources. OEMs are being pushed to provide more complex functionality. This ranges from drivers for new devices and interfaces to more sophisticated, intuitive, and consistent user experiences across product lines and generations, as well as personalized support and services based on consumer actions. The result is a shift in how organizations hire and deploy some of their most valuable assets: the developers in their engineering organizations.

How the Choice of Operating System Supports a Platform Strategy

As OEMs become software-driven organizations, the choice of operating system (OS) becomes a critical factor for success. The OS architecture should support development of an intellectual property (IP) library that can be reused across the p roduct portfolio and across generations of products. It must also allow code for drivers and applications to be easily added or updated without requiring the addition of new items to the kernel and a kernel rebuild. With those characteristics, software designers can develop once and share common low-level driver software components (file system, user interfaces, or standard device drivers such as USB) across products. This ability to share or re-use components speeds the introduction of new technologies or out-of-the-box components into as many products as possible, as quickly and efficiently as possible.

Once a decision is made to standardize on a single microprocessor family, moving to a commercial OS becomes the next key decision for a successful platform strategy. With this approach, OS development, updates, and security patches are offloaded to the OS vendor, reducing the need for difficult, time-intensive, and low-value in-house maintenance of the OS. OEMs can keep their most experienced development resources focused on innovating for competitive advantage and preparing for the next wave of consumer demands. Too often, those valuable resources are focused on OS and driver maintenance, which provides no added or competitive value but is complex enough to require senior-level resources. Plus, without a common OS deployed across the organization, each project or product team will require a senior development resource for OS maintenance, further multiplying the cost and inefficiency.

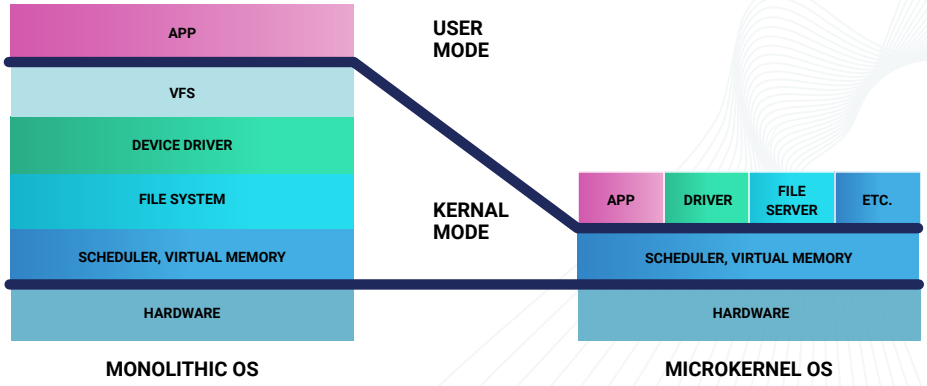


Figure 1: Comparing Monolithic vs. Microkernel OS. In a monolithic OS the drivers, file systems, network stacks, etc. all sit within the kernel space. In a microkernel, these components all run in the user space, so changes to the drivers or file system don't affect the kernel.

Microkernel OS Powers a Scalable Software Platform

Going beyond the need for a standard, commercial operating system, OEMs need a software platform that is scalable, reliable, and secure, and that provides a rich (and familiar) development environment and long-term roadmap, as well as technical support and engineering services. To achieve this, a microkernel architecture offers distinct advantages over traditional monolithic OS architectures, such as Linux®, Android®, and Windows™.

A monolithic OS runs all OS services in the kernel space—including all the drivers, file system, multimedia stack, and network stack. This design means that developers who need to update or add new driver or system code must modify and rebuild the entire kernel, thus creating a unique kernel for each product. Having a unique kernel for each product multiplies the already-intensive testing effort required to release a custom kernel. This slows down NPI capacity and can grind the development process to a standstill. Commonality at this layer of the software stack is critical to scaling your entire organization—engineering, testing, customer experience, etc.

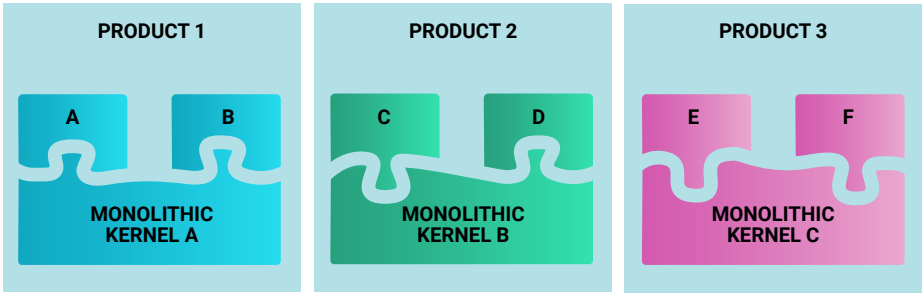


Figure 2: In a monolithic OS, each product has a unique monolithic kernel that includes drivers, file systems, stacks, etc (A, B). Unique monolithic kernels must be developed and maintained for each product.

A perfect example of a monolithic kernel spawning custom kernels is when organizations use Linux for their OS. Many companies using Linux end up with many different OS variants. In a Linux environment, each driver or software component sits within the monolithic kernel, and each component must be integrated, compiled, and tested on each OS. If every product needs a file system or network stack, extensive work is needed to integrate that component into each OS variation—it doesn't scale in individual products or across product lines. This slows the addition of new features and makes it difficult to use standard third-party components. Large companies with multiple product lines may end up with tens or even hundreds of different OS variants, creating an exponential nightmare when OS updates are required or new features need to be added.

Unlike a monolithic kernel, a microkernel separates file systems and the multimedia and network stacks so that they run in user space. This means that a microkernel OS rarely needs to be recompiled and component integration can be done once and reused across multiple products. This separation enables the critical innovations of modularity and scalability. As well, if the same microkernel OS is used across multiple products, device drivers and connectivity protocols can be shared as applications, eliminating the need for custom integrations. Developers can easily reuse custom code or drivers, bring up new hardware, or add off-the-shelf, third-party components to all products quickly. By standardizing on a commercial microkernel OS, OEMs no longer need to deploy valuable resources to maintain the critical underlying functionality. Patches and updates are developed and managed by OS experts, allowing new applications and features to be rolled out in multiple products with minimal effort.

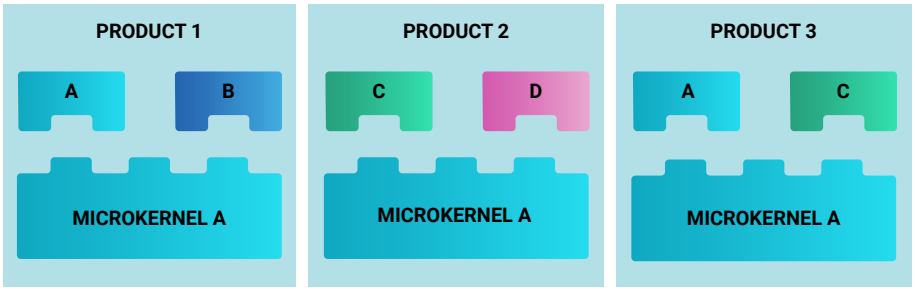


Figure 3: In a microkernel OS, all drivers, network stacks, and other OS services sit outside of kernel space. This means components and code can be reused across products and updates to them do not impact the kernel.

From a hardware development standpoint, a microkernel OS is inherently modular and scalable, which allows developers to easily port code up and down the scale of processors for different product needs. Most commercial OS have a wide range of board support packages (BSPs) available to help get up and running quickly on new hardware, in hours rather than weeks.

A commercial microkernel OS is the basis for engineering efficiency and scalability, especially within large OEMs that have multiple development teams, markets, and product lines. For OEMs who want to be more agile and responsive to changing client demands or market opportunities, choosing a standard microkernel OS for all products and product lines can remove significant pressure from already-stressed development teams, and allow developers to focus on critical innovations instead of maintenance.

The Advantages of a Platform Strategy

Accelerate time to market for new products

OEMs can quickly develop new products by leveraging the code base from other products and developing only software needed to integrate unique components or hardware. This is especially true if the OS vendor possesses a rich set of BSPs so new hardware can be brought up quickly. With a microkernel OS and a shared library of software components, OEMs can increase NPI volume and reduce time to market with the same number of developers.

Reduce hardware supply chain risk

Selecting a standard processor architecture such as Arm or Intel x86, OEMs are no longer at the mercy of a single board provider. With a commercial microkernel OS and BSPs, OEMs can swap hardware as needed to address cost concerns or supply chain interruptions without having to start from scratch for every project.

Drive flexible capabilities across product lines and teams

A standardized platform enables an internal development community to share drivers and other applications, resulting in faster product development across product lines and teams worldwide. The common set of tools enables cross-business technology development and flexibility that was previously impossible. This also increases customer satisfaction, with familiar capabilities and interfaces across product lines.

Focus on competitive product roadmaps and new revenue streams

With the OS managed by outside experts, OEMs can focus their most valuable engineering resources on developing competitive new features such as cloud-based services, enhanced security, or Internet-of-Things (IoT) protocols that differentiate products or meet emerging customer demands. OEMs gain the ability to redeploy high-level developers to new software initiatives such as cloud-based analytics—services that drive new revenue streams.

Increase development velocity without increasing headcount

By depending on the OS vendor to maintain and upgrade the OS platform, OEMs improve the efficiency of the development cycle end-to-end. OEMs can release more products with the same development staff and deliver more value with the same resources.

Create a consistent user experience

Today's consumers expect intuitive, smartphone-like interfaces on the devices they interact with. For many OEMs, user experience (UX) has become a strong differentiator. With a microkernel OS, a common UX can more easily be provided across product lines to provide consistency and a strong brand experience, even across a large product portfolio.

The Best Software Foundation for a Successful Platform Strategy

When looking for a commercial microkernel OS, OEMs around the world turn to BlackBerry® QNX®. The QNX Neutrino® Real-time Operating System is trusted in millions of products across critical automotive, medical, robotics, transportation, military, and industrial embedded systems. The BlackBerry QNX RTOS is POSIX-compliant, making it one of the industry's easiest operating systems to port to.

In addition, developers ramp up quickly on the QNX OS, as it looks and feels like Linux, and uses the same tools. Development teams can design and build systems using standards-based tools (e.g., GCC toolchain, Eclipse IDE) and APIs (e.g., PSE54, Linux, OpenGL ES), while leveraging trusted foundational software that scales from single and multi-core to high-performance compute platforms to ensure maximum portability and design flexibility.

BlackBerry QNX offers an extensive range of board support packages, as well as professional services and service packages such as porting and architecture assessments, to help streamline development timelines. While moving from an open source OS to a commercial microkernel OS increases upfront licensing costs, it significantly decreases other costs, resulting in a strong return on investment.

Platform Standardization Case Study: Zebra Technologies

Zebra Technologies is a global OEM with a wide portfolio of products that range from barcode printing to mobile computing, data capture, locationing and data platforms, as well as related software, services and supplies. In Zebra's specialty printing group, Victor Salmons, Vice President of New Product Development, leads a team of engineers that design enterprise-level, mission-critical thermal printing products for healthcare, manufacturing, transportation, logistics and retail applications.

The group's three specialty printer categories—acquired through mergers and acquisitions—each had a different OS, or no OS at all. This situation made it difficult to deliver a common user experience to customers and to integrate, deploy and manage those products over time. This was frustrating for customers and was also highly inefficient from a development standpoint. When customers asked Zebra to add USB host capabilities to its products, the company realized that there was no easy way to add standard third-party drivers without extensive custom integration and testing. It was time for a new approach.

About 10 years ago, Zebra moved to a standard, scalable platform architecture based on a family of Arm processors and the QNX Neutrino RTOS. With that platform in place, Zebra developers could now build features and functionality from an ecosystem perspective. All of the applications work across the entire portfolio of more than 50 product models and derivatives. The company has also been able to drive a successful initiative around cloud analytics, which allows products to be securely managed and provides data Zebra can use for ongoing product improvements.

Today, Zebra has accelerated its NPI rate from one product per year to four per year, with new features and functions easily added to existing products. This dramatic improvement occurred without increasing software or firmware headcount. Zebra was able to redeploy valuable developers who were working on OS maintenance to focus on value-add features and components. With the foundation in place, product roadmaps can be developed well in advance to identify needed features. Because drivers and components sit outside of the kernel, the microkernel doesn't change, and by leveraging BlackBerry QNX's extensive BSP library, new hardware or features can be brought up in hours or days rather than weeks.

When is the Right Time to Switch to a Platform Strategy?

Ultimately, customer demands for easier setup and manageability, increased functionality and greater security drive OEMs to a platform strategy. If OEMs don't make the shift to a scalable, standard architecture platform, they can't continue to meet market demands and their businesses will stagnate. In the meantime, their competitors may be making the tough changes that are necessary to thrive.

When is the right time to change to a scalable development platform? Whenever business as usual is no longer feasible. That may be in the early stages of a strategic new product introduction; before, during, or after an acquisition; or in response to a strong competitive threat. Whenever the time is right, proven approaches can guide the transformation. And the initial time investment could be recouped as soon as the second product introduction, based on the experience of BlackBerry QNX customers.

BlackBerry QNX provides an ideal software foundation that helps companies bring more new products to market more quickly, frees developers to focus on value-added product features, and reduces time spent on system maintenance. OEMs with multiple product lines and development teams owe it to themselves to investigate the significant benefits of platform standardization.

Contact BlackBerry QNX to learn more.

About BlackBerry® QNX®

BlackBerry QNX is a leading supplier of safe, secure, and trusted operating systems, middleware, development tools, and engineering services for mission-critical embedded systems. BlackBerry QNX helps customers develop and deliver complex and connected next-generation systems on time. QNX technology is trusted in over 175 million vehicles and more than a hundred million embedded systems in medical, industrial automation, energy, and defense and aerospace markets. Founded in 1980, BlackBerry QNX is headquartered in Ottawa, Canada and was acquired by BlackBerry in 2010.