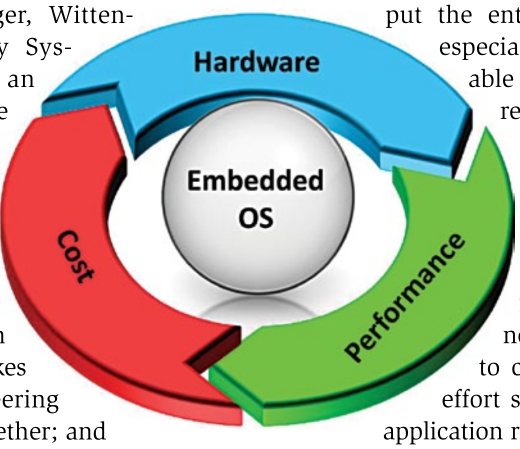
Selecting The Right Operating System for Your Next Embedded Design

Abhishek A. Mutha is a senior technical correspondent at EFY

From robots, cars, home appliances to calculators, thermostats, ATMs and mobile phones, embedded systems are everywhere. And at the heart of almost every embedded system is its operating system (OS), which plays a critical role in keeping the system alive and running. Therefore it is essential to choose the right OS at the very beginning of the design cycle itself. Let us explore the important parameters that must be considered while selecting the perfect OS for your embedded application.

An OS can greatly affect the development of the design. According to Andrew Longhurst, engineering and business development manager, Wittenstein High Integrity Systems, by selecting an appropriate OS, the developer gains three things: one, a task based design that enhances modularity, simplifies testing and encourages code re-use; two, an environment that makes it easier for engineering teams to develop together; and three, abstraction of timing behaviour from functional behaviour, which should result in a smaller code size and more efficient use of available resources.



Peripheral support, memory usage and real-time capability are key features that govern the suitability of the OS. Longhurst says, “Using the wrong OS, particularly one that does not provide sufficient real-time capability, will severely compromise the design and viability of the final product. The OS needs to be of high quality and easy to use.” He adds, “It is hard enough developing embedded projects and you do not want to be struggling with OS-related problems as well. The OS must be a trusted component that the developer can rely on, supported by in-depth training and good, responsive support.”

In the case of systems with real-time characteristics, a hard, real deterministic OS would be the right choice, whereas for applications that require no real-time behaviour but run a set of applications with rich user experience, an embedded Linux with a good graphics library or Android would be the right fit, informs Thilak Kumar, manager, field engineering, Wind River Systems. Therefore choosing the right OS early in the design cycle is very important. He says, “If it is not realised, it could put the entire project at risk, especially if the OS is unable to meet key system requirements.”

On another note, Mubeen Jukaku, technology head, Emertxe Information Technologies, feels, “Design engineers should be able to create a design with effort spent in creating the application rather than other factors specific to the OS.”

Now that we know the importance of choosing the right OS for developing embedded systems, let us take a look at the parameters to be considered for selecting the same.

Selecting the OS

Embedded systems are meant to run for long, and sometimes these are unattended or non-upgradable. In any case, these should be robust, reliable and secure. “Support for device drivers, ease of porting and extending/configuring the kernel also matters a lot if devices are peripheral-rich

It is hard enough developing embedded projects and you do not want to be struggling with OS-related problems as well. The OS must be a trusted component that the developer can rely on, supported by in-depth training and good, responsive support

and you have future plans of upgrading the hardware," says Jukaku. For power management, the OS should be able to provide power-saving features, like suspend/wake-on-interrupts. He adds, "Some other factors include availability of software protocols and development libraries, which could be specific to the application area. The level of vendor/community support for the OS also needs to be high."

While designing an embedded system, parameters such as computing power, memory, electrical power, real-time behaviour, regulatory guidelines, connectivity, safety, security and manageability should also be considered while selecting an OS. Citing an example, Kumar says, "If you are designing a life-critical device, such as a pacemaker, then the OS would need to be deterministic, small and extremely power efficient." He adds, "If you are designing an avionics system, it would still need to be power efficient but not as much as the pacemaker. For sub-systems, meeting safety requirements outlined by the regulatory authority is one of the most important requirements and a certified/certifiable OS would be more appropriate."

For developers, real-time operating system (RTOS) selection has traditionally been a matter of preference and convenience, as they tend to look at compatibility with their choice of compilers, debuggers and other development tools, informs Prasad Suri, AVP-sales, product engineering services, ValueLabs. He says, "Many use integrated development environments (IDEs) that enable them to develop a wider range of RTOSs."

Another critical factor for the success of a project is the selection of an OS that ensures right time to market for the application. Suri adds, "RTOSes that offers simple system services, intuitive naming conventions, documentation, good support and availability of full source code should be preferred as these characteristics enable developers to become productive in a short period and complete projects on schedule."

Important parameters to consider before finalising an OS

Responsiveness. The OS-scheduling algorithm, interrupt latency and context switch times will significantly define the responsiveness and determinism of the system. The most important consideration is the kind of response desired (Is a hard real-time response required?). This means that there are precisely-defined deadlines that, if not met, result in system failure. Alternatively, would a non-deterministic, soft real-time response be appropriate? In this case, there are no guarantees to when each task will be completed.

Available system resources. Micro-kernels are designed to work using minimum system resources and provide limited but essential task-scheduling functionality. Micro-kernels generally deliver a hard real-time response, and are used extensively with embedded microprocessors with limited random access memory (RAM)/read only memory (ROM) capacity, but can also be appropriate for larger embedded processor systems.

Alternatively, a full-featured OS, like Linux or WinCE, could be used. These provide a feature-rich OS environment, normally supplied with drivers, graphical user interfaces (GUIs) and middle-ware components. Full-featured OSs are generally less responsive, require more memory and more processing power than micro-kernels, and are mainly used on powerful embedded processors where system resources are plentiful.

Middle-ware and drivers. The OS supplier should be able to provide the middle-ware and drivers you require, integrated with the OS and your target hardware. The vendor should also be in a position to provide support and ultimately take responsibility for the complete package.

Open source or professionally-licensed. There are many widely-used, free open source OSs available, distributed under general-public licence (GPL) or modified-GPL licences. However, these licences may contain copy-left restrictions and offer little protection. Professionally-licensed products remove the copy-left restrictions, offer full intellectual property (IP) infringement indemnification and warranties. In addition, you have a single company providing support and taking responsibility for the quality of the product.

Cost. A proprietary OS results in increased cost. For instance, Windows-enabled phones require licence pay-outs to Microsoft, thereby pushing up the cost of the product. Open source is free. Many companies have customised open source to a high degree and charge premium for further customisation.

Functionality levels. In terms of functions, permitting back and forth data transfer rather than only one-way would be beneficial for your embedded design. Your design is further benefited if the OS provides dynamics during run-time, supports functions to handle interrupt requests, handles multiple tasks, supports multiple platforms/architecture and is highly stable and secure. If not, it significantly increases the development time and resources required.

Flexibility. A proprietary OS licence will not permit actual alteration of the OS itself, whereas open source OSs are open to complete alteration as per the need of the hour, as many times as required.

Quality. What emphasis does the OS supplier place on quality within his or her organisation? Quality is more than just a coding standard. Are correct procedures in place to guarantee the quality of future products and support? Well-managed companies that take quality seriously tend to be ISO 9001 certified.

Safety certification. Pre-certified and certifiable OSs are available for applications that require certification to international design standards, such as DO-178C and IEC 61508. These OSs provide key safety features, and the design evidence required by certification bodies to confirm that the process used to develop the OS meets the relevant design standard.

—Contributions from Andrew Longhurst, engineering and business development manager, Wittenstein High Integrity Systems, and Neeraj Saraf, CEO, Seal Technologies

Point of view: Why Linux as an OS?

Linux is the primary choice for embedded systems developed at Emertxe because it is very robust, reliable, secure and scalable. It is a highly configurable, open source OS with plenty of development tools and application packages available. Board-support packages (BSP) and drivers are available for many platforms in the vanilla kernel. Adding support for new platforms/devices is also easy. You get community support, which is often better than vendor support.

With this, the focus can be on building the application without getting concerned about the functionality of the underlying OS.

—Mubeen Jukaku, technology head, Emertxe Information Technologies

MAJOR CONTRIBUTORS TO THIS REPORT

Andrew Longhurst engineering and business development manager, Wittenstein High Integrity Systems	**Mubeen Jukaku** technology head, Emertxe Information Technologies	**Neeraj Saraf** chief executive officer, Seal Technologies	**Prasad Suri** AVP-sales, product engineering services, ValueLabs	**Thilak Kumar** manager, field engineering, Wind River Systems

Trends in selection of an OS

"We have seen an increase in the use of multi-core devices," notes Longhurst. This presents an interesting challenge to OS suppliers, as the OS also needs to support core-to-core communication and asymmetrical and/or symmetrical processing models. He adds, "The type of OS support required is highly dependent upon the architecture of the application. Therefore a one-size-fits-all approach is not appropriate, as each solution will require a certain amount of customisation to achieve an optimum design."

Another obvious trend is related to the Internet of Things (IoT) or machine-to-machine (M2M) communication, where embedded devices that existed in isolation in the past need to be connected now. Kumar says, "Connectivity is essential for better manageability of assets, which allows businesses to move from a device-centric model to a service-oriented model. With connectivity, there is also the threat of security that needs to be addressed."

The other very prominent trend is software defined networking (SDN) and network function virtualisation (NFV). He adds, "This is driving consolidation of efforts in the networking and telecommunication markets where delivering carrier-grade reliability, while also achieving high-performance throughput with minimal latency, is absolutely essential."

Platform era. There has always been a need to tailor-make embedded OSes for specific application domains. Many times this is like re-inventing the wheel and often unnecessary. Jukaku notes, "Recently, there has been a trend in building domain-specific OSes or software stacks that consist of the OS, application stack, framework and development environments—commonly known as platform." Citing an example, he explains, "In the automotive industry, there is automotive-grade Linux, which is a Linux based software stack for the connected car. Google is also bringing Android to the car with Android auto. Similarly, in the IoT space, ARM has come up with mbed OS for IoT devices."

Adoption of open source. Another trend seen is the wide adoption of open source software in the embedded space. Jukaku says, "Organisations are adopting open source software because of their reliability, stability, accuracy, cost, openness and support."

If you are designing a life-critical device, such as a pace maker, then it would need to be deterministic, small and extremely power efficient. In this case, a highly deterministic operating system that can run with minimal computing power and memory is appropriate

Choose wisely, build effectively

It is not only the OS functionality and features that you will need to consider, but also the licensing model that will work best for your project's budget and the company's return on investment. Longhurst says, "The company behind the OS is just as important as selecting the correct OS itself." He adds, "Ideally you want to build a relationship with the OS supplier that can support not only your current product but also products of the future. To do this, you need to select a proactive supplier with a good reputation, working with leading silicon manufacturers to ensure they can support the latest processors and tools." Trust, quality of product and quality of support is everything.

From skill point of view, Jukaku says, "Understanding the architecture of the OS, integrating appropriate board-support packages, hardware interfacing and customising, and tuning for specific needs are all very important to have." These skills make the OS easier for a new product designer to get started with. ●