

Nucleus OSEK: Real-time kernel for the automotive industry based on Nucleus PLUS

Nucleus OSEK is Accelerated Technology's full-featured implementation of the OSEK/VDX Operating System. (The Communication and Network Management modules are additional products available to support Nucleus OSEK.) Nucleus OSEK was developed specifically for embedded applications in the automotive industry. Care has been taken to produce a product that is fully compliant with the OSEK specification version 2.0. Because of Accelerated Technology's existing source code, no royalty business model, and the way current Nucleus products are developed, it made sense to become an early participant in OSEK technology.

ABOUT OSEK

OSEK/VDX is a joint project of the automotive industry. It aims for an industry standard of an open-ended architecture for distributed units in vehicles. A real-time operating system, software interfaces and functions for communication and network management tasks are thus jointly specified.

The term OSEK means "Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug" (Open systems and the corresponding interfaces for automotive electronics). This particular standard originated in Germany, and included BMW, Bosch, Daimler-Benz, Opel, Siemens, VW, and the IIT of the University of Karlsruhe. The term VDX means "Vehicle Distributed eXecutive." This standard originated in France and included PSA and Renault.

The functionality of the OSEK operating system was

harmonized with VDX. For simplicity, OSEK will be used instead of OSEK/VDX. Part of the motivation behind the development of OSEK requirements was high, recurrent expenses in development, irregular management of non-application related aspects of control unit software, and the incompatibility of control units made by different manufacturers due to different software interfaces and protocols.

There are three primary modules that comprise the OSEK architecture:

- Operating System. A real-time executive for Electronic Control Unit (ECU) software and the basis for the other two OSEK modules.
- Communication. The data exchange within and between Control Units.
- Network Management. Configuration determination and monitoring.

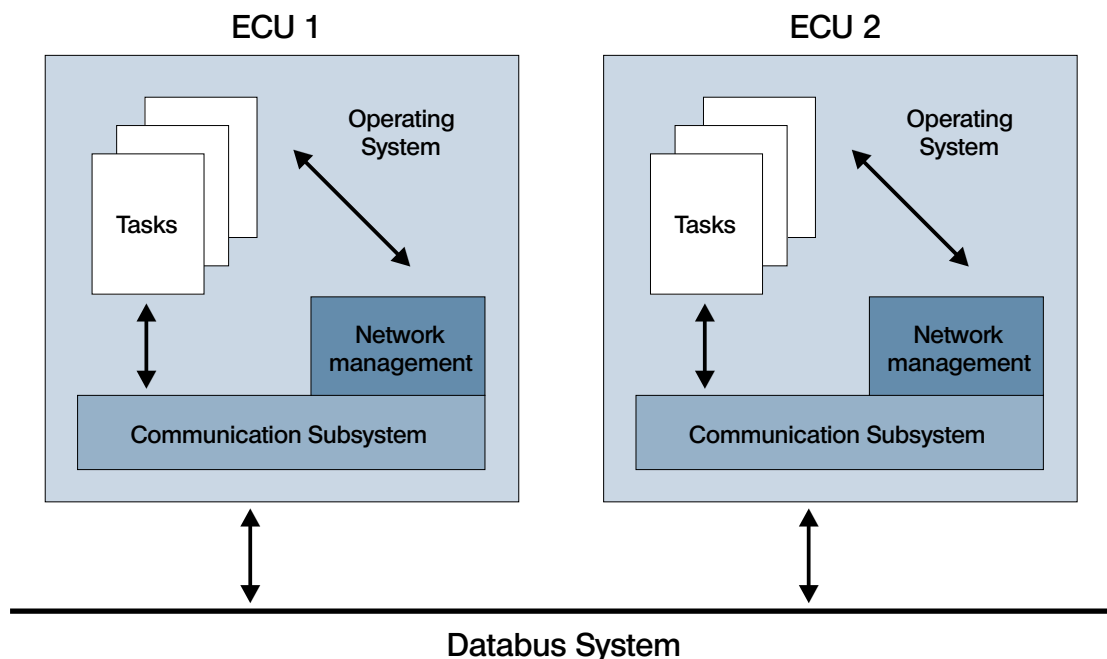


Figure 1.

Figure 1 is a graphical representation of the OSEK Architecture.

Requirements for the OSEK Operating System include strict real-time requirements. Memory resource usage must be low-level, and the operating system must be scaleable, reliable, ROMable, and cost-sensitive. The software must be developed in a manner to promote portability. It must serve as the basis for software integration from various manufacturers. Finally, the configuration, including scalability and scheduling policies, must be static.

The goals of the OSEK project are to support the portability and reusability of the application software by providing specifications for interfaces that are abstract and as application-independent as possible. Additionally, the specifications of a user interface should be independent of supplied hardware and network applications. The architecture will also be more efficiently designed because of the guidelines. The functionality shall be configurable and scaleable, to enable optimal adjustment of the architecture to the application in question. Finally, verification of the functionality and implementation of prototypes in selected pilot projects shall be available.

ABOUT NUCLEUS OSEK

Nucleus OSEK is Accelerated Technology's full-featured implementation of the OSEK/VDX Operating System. (The Communication and Network Management modules are additional products available to support Nucleus OSEK.) Nucleus OSEK was developed specifically for embedded applications in the automotive industry. Care has been taken to produce a product that is fully compliant with the OSEK specification version 2.0. Because of Accelerated Technology's existing source code, no royalty business model, and the way current Nucleus products are developed, it made sense to become an early participant in OSEK technology.

Nucleus OSEK is mostly written in ANSI C and is, therefore, extremely portable. Nucleus OSEK is typically implemented as a C library. Real-time Nucleus OSEK applications are linked with the Nucleus OSEK library,

and the resulting object may be downloaded to the target or placed in ROM.

The benefits of Nucleus OSEK are numerous, including reduced costs and shorter time to market. The quality of the software in the control units of all companies is enhanced by adherence to the OSEK standards. Interfacing features for control units with different architectural designs are standardized by the use of Nucleus OSEK. Sequenced utilization of the existing resources that are distributed in the vehicle enhance the overall system's performance without requiring additional hardware.

As all system objects will be defined statically; a generation tool for system generation is provided. (See Figure 2)

Nucleus OSEK, like all Nucleus products, is provided to customers in source code, and no royalties are charged.

CHARACTERISTICS

Task Types

There are two task types required for OSEK functionality:

- Basic tasks only release the processor if
 - They are being terminated,
 - The system is executing higher priority tasks, or
 - Interrupts occur with cause the processor to switch to an interrupt service routine.
- Extended tasks can additionally have a waiting state, when they are using the event mechanism for synchronization.

Conformance Classes

The conformance classes will provide convenient groups of operating system features for easier understanding and discussion of Nucleus OSEK. The conformance classes will allow partial implementations, which may be certified as OSEK-compliant, along pre-defined lines. They will also create an upgrade path from classes of less functionality to classes of higher

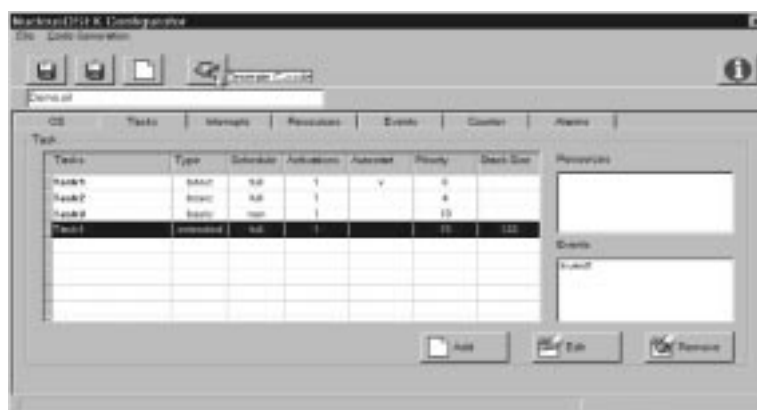


Figure 2. Nucleus OSEK Configurator.

functionality with no changes to the application using OSEK related features.

There are four conformance classes defined for OSEK functionality:

- BCC1. Only basic tasks, limited to one request per task and one task per priority, while all tasks have different priorities.
- BCC2. BCC1 functionality, plus more than one task per priority possible and multiple requesting of task activation allowed.
- ECC1. BCC1 functionality plus extended tasks.
- ECC2. BCC2 functionality plus extended tasks, multiple requesting of task activation only for basic tasks allowed.

Nucleus OSEK will support all four conformance classes.

Scheduling Policy

Three scheduling methods are defined for OSEK functionality:

- **Non-preemptive:** A task switch is only performed via one of a selection of explicitly defined system services. Non-preemptive scheduling imposes particular constraints on the possible timing requirements of tasks. Specifically, the non-preemptable section of a running task with lower priority delays the start of a task with higher priority up to the next point of rescheduling.
- **Full-preemptive:** A task which is presently running may be rescheduled at any instruction by the occurrence of trigger conditions pre-set by the operating system. Restrictions are related to the increased memory space required for saving the context, and the enhanced complexity of features necessary for synchronization between tasks. Access to data, which are used jointly with other tasks, must be synchronized.
- **Mixed-preemptive:** A mixture of non-preemptively and full-preemptively scheduled tasks.

Because the non-preemptive and full-preemptive scheduling mechanisms are basically subsets of the mixed-preemptive scheduling method with minor variances, the initial release of Nucleus OSEK will only include the mixed-preemptive scheduling method.

Later releases of Nucleus OSEK will support non- and full-preemptive methods as well.

Synchronization Mechanisms

There are two synchronization mechanisms defined for OSEK functionality:

- Resource management: Controls access to jointly used logic resources or devices. Program flow control.
- Event management: Event management for task synchronization, allowed only for extended tasks.

Nucleus OSEK will support both synchronization mechanisms.

ALARMS

There are two types of alarms defined for OSEK functionality:

- Relative: Count values are defined relative to the actual counter value.
- Absolute: Count values are defined as absolute values.

The count values can be defined dynamically at runtime (variant alarms) or statically at compile time (non-variant alarms.)

Nucleus OSEK will support relative and absolute alarms in the variant manner. Because non-variant alarms are not a "hard" requirement, we do not feel they make sense by the software concept.

ERROR TREATMENT

Nucleus OSEK will utilize user-definable Hook Routines for error handling and debugging.

Scalability of error checking will include an extended status for development phase and a standard status for production phase ■

Lothar Bauer, Junior Manager of Epsilon, was born on June, 27th 1953 in Lohr/Germany. He studied "Computer Science" at the Technical University in Darmstadt/Germany and has 18 years of experience in developing embedded systems for printers, scanners, car communication systems, and medical devices. Since 1996, he has been Junior Manager and partner of Epsilon Incorporated.