Recent development in operating system based on AUTOSAR standard used in ECUs for automotive vehicles

Purva P. Gokhale¹, Maruti Jadhav²

¹²Dept. of Electronics and Telecommunication, Rajarambapu Institute of Technology, Rajaramnagar, Maharashtra, India

e-mail: purvagokhale1996@gmail.com

Abstract. Development in automotive embedded systems has grown rapidly in recent years due to increased functionality in automotive vehicles. Nowadays, a lot of modern technologies have been implemented in automotive vehicles in order to improve the performance and reliability. Automotive Open System Architecture (AUTOSAR) is a software standard used for development of the new modules inside of the automotive Systems. The evolution in automotive vehicles impacted the usage and design of real time operating systems as automotive embedded systems are mainly dependent on real time operating systems used in ECUs inside automotive vehicle. AUTOSAR based operating system is also a real time operating system which is developed on the basis of OSEK (Open Systems and the Corresponding Interfaces for Automotive Electronics) standards. AUTOSAR is nothing but a standardized extension of OSEK and both play very important role in software development of automotive systems. In this paper, the recent trends in the development in automotive embedded system have been discussed. The standards required for designing a real time operating system have been briefly explained for better understanding of the developments in the automotive embedded systems

1. INTRODUCTION

In automotive vehicles, different functionalities are provided by multiple software tasks running on individual microcontroller with the help of ECUs (Electronic control units). Also operating system supports microcontroller to run multiple software tasks as per the real time requirements. Hence, increase in functionalities provided by automotive vehicles also increase the number of ECUs used in automotive embedded system [1]. As a result, automotive embedded systems have become much complex and strongly integrated. The real time operating systems are also facing new challenges related to safety criticality and timing constraints due to increase in complexity of automotive systems. Hence, automotive industries are implementing many new technologies in automotive software. Due to these new technological implementations many research opportunities have generated in AUTOSAR field but the academic students and researchers are not so aware of it. Hence, this paper may help to bridge the gap between the growing development in AUTOSAR and the academic world.

In order to compete with the growing complexity of automotive electronics and software architecture inside a car, OSEK and its extension AUTOSAR standards were used by many car manufacturers. These standards helped to exchange or upgrade any software or hardware architecture
used inside automotive electronic system. These standards improve the performance, safety and cost efficiency of automotive systems without compromising with the quality. The automotive standards used for the development of the automotive embedded systems have been briefly explained in the following sections.

1.1 OSEK (Open Systems and the corresponding interfaces for Automotive Electronics)

The main specification of OSEK operating system is to provide uniform environment which supports efficient utilization of resources used in automotive electronic control unit software. OSEK is a short form of German term, “Offene Systeme und deren Schnittstellen fuer die Elektronik im Kraftfahrzeug”, and translated as “Open Systems and the Corresponding Interfaces for Automotive Electronics”. OSEK is also known as VDX (Vehicle Distributed Executive). The services provided by OSEK are briefly described below [2].

- Management of Tasks - Task is classified into two types, extended and basic task. Both of the types have three task states which are Running, suspended and ready state but basic difference between both of the types is extended tasks have an additional state called as wait state as shown in Figure 1. Wait state releases the processor and shifts to a low priority task without suspending the currently running extended task. The purpose of each state in basic task has given below:

  Running: Task comes in this state when it gets processor to perform its operation.

  Ready: Task comes in ready state when it gets preempted by any other high priority task or Isr. Also if any already suspended task gets a resource or processor to perform its process then it will come to ready state.

  Suspended: If any task completes its execution or terminates then it comes to suspended state. One additional task state has added in the extended task state model. Extended task allows processor to run system calls or application program interface (API) routines. While executing the system call or API routines, processor keeps the currently running task in waiting state.

  Waiting: When system call arrives, processor stores all the contents of running task into context saving registers and performs a system call. After completion of system call or API routine process, processor releases all data related to that completed procedure and restores the saved contents of previous task and comes in ready state again.

- Scheduling policy - In OSEK, scheduling is done before code starts running which means that all tasks are statically configured. Scheduling is done on the basis of priorities assigned to tasks. Priorities are of two types, preemptive and non-preemptive priority. One currently running task is having preemptive priority and if any high priority task preempts it, then processor suspends running task and starts high priority task. While any non-preemptive priority task does not allow any high priority task to preempt it.

![Figure 1. Task State Model](image-url)
• Interrupt service routine (Isr) management- OSEK provides two categories of interrupts, CAT1 (category 1) and CAT2 (category 2). CAT1 interrupt allows processor to have an API call (Application Program Interface) or system call. API is nothing but a set of service routines used for developing software and application. To use API, programmer requests processor through API or system call. CAT2 interrupt does not allow such API or system call during its execution.

• Management of resources – when two different tasks are trying to get one resource then processor goes in deadlock. Deadlock is a situation in which each process is taking a resource and waiting for another resource which is already occupied by some other process. To avoid such situations OSEK uses a special protocol known as PCP (Priority Ceiling Protocol) [3]. PCP works under principle of priority inversion. Three types of priorities are present in PCP, high (H), medium (M) and low (L). If any low priority task occupies any resource then PCP will increase that task’s priority to high or medium according to scheduling of task so that any other high priority task will access its resource. This protocol resolves much load on processor so that it will work smoothly and increase the system efficiency.

• Alarms (timers and counters) - The OSEK operating system gives services for processing recurring events. These events may be of the type where timer is providing an interrupt at regular intervals of time or interrupt is occurring when counter reaches to its predefined value. Here occurrence of interrupt is nothing but start or expiry of an alarm.

• Message communication- In OSEK, messages is used for communication within processor. Message communication in OSEK is described in OSEK COM specification.

• Error handling and Hook routines - When processor switches from one task to another task or from task to Isr or from Isr to task, there may be possibility of occurrence of error. Error may cause due to occurrence of linking problem while switching from task to task or Isr to task or vice versa. Hook routine is basically used to join two processes. To handle such kind of errors, OSEK offers hook routine mechanism. Hook mechanism provides different types of hooks- startup, shutdown, pre-task, post-task, stack fault and error hook [4]. These hook routines can take of starting, ending as well as switching of tasks or Isr.

1.2. AUTOSAR (Automotive Open System Architecture)

AUTOSAR divides an ECU into standardized components and software layers. AUTOSAR is used for creating and establishing open and standardized layered software architecture for an automotive ECU [5]. As AUTOSAR is a standardized extension of OSEK, it also provides different features mentioned below.

• Reusable- AUTOSAR is a standard for software development. So that it has reusable architecture. Hence we can add different functionalities in automotive vehicles without doing huge change in architecture. Because of reusability, system development with the help of AUTOSAR is cost effective.

• Timing protection through Scheduling policy- Scheduling policy in AUTOSAR contains a mechanism called highest priority first. In highest priority first policy, tasks having same priority are grouped together and placed in the FIFO stack. These all tasks have a shared resource between them. One task among the group occupies a resource and locks it so that another task does not take that resource. When that task completes its operation and unlocks or releases the resource then only the successive task in the FIFO stack will occupy the resource released by the previous task. Also one task does not preempt another task within a group made on the basis of priority level. These all preemption rules provide timing protection to system so that system can meet all the timing requirements.

• Alarms and counters-Alarms are nothing but the signals provide by mechanical modules inside engine of a car. Starting and expiring of alarm is dependent on the counter ticks. Alarms are driven by interrupts present inside a microcontroller present at the lower most layer of AUTOSAR architecture.

• Schedule Tables- Schedule table is just like alarm with an extra feature. Schedule table is also dependent on counter. Schedule table contains a set of expiry points. When processor reaches to an expiry point, one or more processes such as activation of task or setting of an event have carried out. When any task has activated on expiry point in schedule table then that task will run till its completion without preemption.
2. RECENT DEVELOPMENT IN AUTOMOTIVE EMBEDDED SYSTEM

Earlier automotive embedded systems were having a single processor with single application running on it. Also those systems were having high power consumption with low efficiency. In order to improve efficiency of embedded systems, multiple tasks should run simultaneously on single processor.

In recent years, most of car manufacturers are using multicore processors in automotive embedded systems to increase efficiency. Multicore processor has two or more processing units inside it. Hence multicore processor can run multiple tasks on single processor at same time. Also, more new functionalities have developed in automobiles such as vehicle to vehicle (V2V) communication, vehicle infotainment system (IVI) as well as vehicle navigation system. For implementing those functionalities in vehicle, automotive system inside vehicle should satisfy the requirements like wireless Bluetooth and internet technology as well as global positioning system (GPS). To fulfill such kind of requirements high performance hardware such as multicore processor is required [6].

Implementation of new functionality inside vehicle also affected the design of embedded system. Due to development in vehicle functionality, now few more specifications are considered for design of automotive embedded systems such as Complexity, Flexibility, Dependability, Connectivity, Maintainability, Upgradeability, Size and Power Consumption [7]. These specifications are briefly described as follows-

- **Complexity**- It depends on the features provided by embedded system. These features impact the code size of embedded system. As a result design complexity increases.
- **Flexibility**- It depends on the software setup required for implementation of new changes in design.
- **Dependability**- As many OS applications have integrated with the software architecture, design is dependent on the integrity level of software architecture.
- **Connectivity**- To have new technologies in vehicles, automotive ECUs must be connected to internet, Bluetooth, GPS system. Hence, connectivity is the main concept of design.
- **Maintainability**- Design of automotive embedded system should be maintainable as it is used in automotive vehicles.
- **Upgradeability**- Design should be easy to upgrade. If some new functionality has added in automotive system then we have to change the design of that system. Hence, it should not take more time to implement new functionality in current design.
- **Size and Power Consumption**- These are the crucial specifications used for design of automotive embedded systems. Size of system should be small as it is used in automotive vehicles and also it should consume less amount of power.

Hence addition of new features inside a car affects design of automotive embedded system along with design specifications. Design of embedded system is nothing but software architecture of it. Also, addition of new features or functionality also changes or improves the use and design of automotive system development standards.

3. MODIFICATION DONE IN AUTOMOTIVE STANDARDS ACCORDING TO RECENT DEVELOPMENT

Car manufacturers widely use two automotive system development standards, OSEK and AUTOSAR [8]. Hence to cope up with the growing technologies inside a car and development in automotive embedded system, some modification has done in these standards. So, it is necessary to modify the design and architecture of both of these automotive standards to make it compatible with growing technologies in automobiles.

**A Modification in OSEK standard**

Now, automobiles are using high performance hardware for integration of different applications inside automotive embedded systems. So to cope up with these new applications, software architecture inside automotive system should perform multitasking. Several tasks share a single processor to
complete their execution is called multitasking. Multitasking helps automotive system to enhance the efficiency as multiple tasks can run at same time.

The important thing in multitasking is switching from one task to another and restoring the previous task contents required for further processing. This whole procedure is called as context switching. In Context switching, lots of data transmits internally within processor. Internal data transmission enhances the processing time of operating system and decreases efficiency. As processing time of context switching is also dependent on the hardware registers, some researchers have used special hardware to increase the performance of context switching. But it is not beneficial to use special hardware as it increases the effective cost of system [9]. Finally, optimization in context switching has implemented by doing some modification in software architecture of OSEK operating system.

Context switching can be improved by changing the scheduling algorithms such as priority scheduling, first come first serve scheduling, round robin scheduling, preemptive and non-preemptive scheduling [10]. It is better to change the state model of task instead of changing the algorithm. It creates more pressure on processor in implementation of new algorithms. Also, new algorithm implementation results in increased design complexity and effective cost. Also, Optimization in context switching has achieved by modification in task state model used in OSEK scheduling policy [11]. Modification done in task state models of basic and extended task used in traditional OSEK has shown in Figure 2. The modification has done only in ready state of task state models of basic and extended task. Ready state is divided into two states, intermediate and initial.

In case of basic task state model, there are two scenarios, one when the task which is in running state completes its execution and enters into the suspended state while the next running state task can enter into the initial state at a same time and another when task which is in running state gets preempted and goes into intermediate state while the next running state task can enter into the initial state at a same time. In both the scenarios, there is no need to save context or information of next running state task as it is already there in initial state.

Also In case of extended task state model, task which is in running state gets preempted and goes into intermediate state while the next running state task which is already in waiting state will also come in intermediate state after getting a set event. In this scenario, there is no need to save context or information of next running state task as it is already there in waiting state.

![Figure 2. Modified Task State Model](image)

Modified task state models have complex structure than the traditional task state models but optimized context switching process of OSEK OS has achieved with the help of improved structure of task state model. In addition to this, implementation of this technique includes few steps with advantage of less context switching time.
**B Modification in AUTOSAR standard**

To make automotive embedded system compatible with the growing technologies inside automotive vehicle, high performance hardware has been used in automotive ECUs. Implementation of high performance hardware like multicore processor changes the design of automotive systems. As AUTOSAR is the basic standard used for development of automotive embedded systems, it is necessary to make some modification in its software architecture according to hardware. Hence, AUTOSAR consortium has introduced a new version called AUTOSAR 4 which is compatible with multi-core automotive embedded systems [12]. New features introduced in AUTOSAR 4 are briefly described below.

- **Inter OS application Communicator (IOC)** - This communicator is mainly used for communication between two different cores within a processor. Physical memory is required to run different OS applications on processor core. So, each core tries to access memory to run different OS applications. Hence, it is necessary to have communication between the cores for accessing memory to process different applications. In this communication, shared memory buffer has been used to access the memory. The shared memory buffer is stored in processor memory and accessible by the sender as well as the receiver. Sender acquires memory to process a certain application and releases it after completion of that application. After releasing the memory, the sender sends a message on the shared memory buffer. With the help of the message sent by the sender, the receiver understands that the memory is free to process an application. Here, the sender and receiver are nothing but different cores of the processor [13].

- **Locatable entities (LEs)** - In a multicore processor, one core is treated as the master core, and the rest are slave cores. Locatable entities are nothing but tasks, alarms, applications, and counters. All LEs are implemented on the master core only. The master core interacts with slave cores through locatable entities. In a multicore processor, hardware only activates the master core, and activation of slave cores has been done by software according to their requirement.

- **Spinlock mechanism** - Spinlock acts like semaphores used in real-time operating systems. Spinlocks are used for synchronization between the tasks running on different cores. When two tasks are trying to access a single resource, then that situation is called mutual exclusion. Spinlocks are mainly used to get rid of mutual exclusion. Spinlock mechanism uses test and set function to access resources. If a task or thread acquires any resource, then that task locks the resource. The test and set function is used to set a Boolean variable as true so that other task or thread can understand that resource has locked by another task. As soon as the task releases its resource, the test and set function changes the value of Boolean variable as false so that another task or thread can recognize the availability of a new resource. Spinlock mechanism overcomes the disadvantages of Priority Ceiling Protocol (PCP) used for resource management in automotive real-time operating system [14].

**4. CONCLUSION:**

A modern automotive vehicle comes with a lot of features and innovations based on the mechanical, electrical, and electronic systems. As a result, car manufacturers are gradually increasing the use of multicore processors inside the automotive ECUs. In the area of software development for multicore systems, it is enhancing pressure on the operating system for number of automated solutions. Also, ECUs with multicore architecture can cover market only when the effective cost of software development is not increasing drastically. The recent development in OSEK and AUTOSAR automotive standards provides an efficient automotive embedded system with less investment cost and time.

**REFERENCES**

[1] Y. Sun and F. Y. Wang, “A design architecture for OSEK/VDX-based vehicular application specific embedded operating systems,” IEEE Intell. Veh. Symp. Proc., vol. 2005, pp. 882–887, 2005.

[2] D. John, “OSEK/VDX history and structure,” IEE Colloq., no. 523, 1998.

[3] O. Rtos and O. Introduction, “Christian Michel Sendis OSEK / VDX Introduction OSEK / VDXTM • Began in French and German automotive industries • Steering committee includes BMW , Daimler Chrysler , Robert Bosch , Siemens , Volkswagen and several other prominent
automotive • Is now the ,” no. March, 2009.

[4] C. Manager, “AUTOSAR Multicore Operating System Implementation for MPC5668G *1,2,” vol. 2, no. 10, 2013.

[5] M. Genero, L. Jiménez, and M. Piattini, “Conceptual Modeling - ER 2000 ontological analysis of taxonomic relationships P226,” 19th Int. Conf. Concept. Model. Salt Lake City, Utah, USA, Oct. 9-12, 2000, no. October, pp. 513–525, 2000.

[6] R. A. Cherian and T. Michael, “Multicore OS Design based on AUTOSAR for MPC5668G,” vol. 3, no. 10, pp. 1276–1280, 2014.

[7] D. Andrews, I. Bate, T. Nolte, C. M. Otero-Perez, and S. M. Petters, “Impact of Embedded Systems on RTOS Use and Design,” Proc. Work. Oper. Syst. Platforms Embed. Real-Time Appl., pp. 13– 20, 2005.

[8] J. Denil, S. Demeyer, P. De Meulenaere, K. Vanstecheelman, and K. Maudens, “Wrapping a real-time operating system with an OSEK compliant interface - A feasibility study,” Proc. 7th Work. Intelli. Solut. Embed. Syst. WISES 2009, no. section II, pp. 157–164, 2009.

[9] C. Li, C. Ding, and K. Shen, “Quantifying the cost of context switch,” Proc. 2007 Work. Exp. Comput. Sci., no. June, pp. 13–14, 2007.

[10] M. K. M. R., “an Improved Approach To Minimize Context Switching in Round Robin Scheduling Algorithm Using Optimization Techniques,” Int. J. Res. Eng. Technol., vol. 3, no. 4, pp. 804–808, 2014.

[11] Z. Wu, H. Li, G. Yang, Z. Gao, and P. Lv, “An improved method of task context switching in OSEK operating system,” Int. J. Pervasive Comput. Commun., vol. 6, no. 2, pp. 179–191, 2010.

[12] K. Devika and R. Syama, “An Overview of AUTOSAR Multicore Operating System Implementation,” Int. J. Innov. Res. Sci. Eng. Technol., vol. Vol. 2, no. Issue 7, pp. 3163–3169, 2013

[13] K. Zubac, “University of Huddersfield Repository ANALYSIS AND EVALUATION OF INTER-CORE COMMUNICATION BASED ON A MULTI-CORE AUTOMOTIVE SOFTWARE AND HARDWARE,” 2016.

[14] S. Afšhar, M. P. W. Verwielen, P. Gai, M. Behnam, and R. J. Bril, “An implementation of the flexible spin-lock model in ERIKA Enterprise on a multi-core platform.”