ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Volume no 11 Issue 1 January 2024

Operating Systems in Engine Control Unit

[1] Prathmesh Bahir, [2] Prashant Andhale, [3] Minal Deshmukh, [4] Shraddha Habbu

[1] [2] [3] [4] Department of Electronics and Telecommunication Engineering, Vishwakarma Institute of Information Technology, Pune, India
Corresponding Author Email: [1] prathmesh.22111230@viit.ac.in, [2] prashant.22110496@viit.ac.in, [3] minal.deshmukh@viit.ac.in, [4] shraddha.habbu@viit.ac.in

Abstract— In the contemporary landscape of automotive technology, the intricate orchestration of vehicle functions heavily relies on the Engine Control Units (ECUs) that meticulously manage and optimize engine performance. The efficacy of these ECUs hinges upon the underlying operating systems, their capacity for real-time responsiveness, unwavering reliability, and deterministic behaviour. This comprehensive research paper delves into the diverse array of operating systems utilized within Engine Control Units, placing a specific emphasis on real-time operating systems (RTOS). Real-time capabilities stand as linchpins in automotive applications, ensuring the punctual and predictable execution of critical tasks. This study embarks on a thorough investigation of several prominent RTOS solutions that dominate this domain. Among these are the Embedded Kernel, Vehicle Distributed Executive, AUTOSAR, FreeRTOS, QNX, Integrity, and Linux fortified with real-time patches. The paper adopts an in-depth analytical approach, seeking to unravel the nuanced strengths, weaknesses, and overall suitability of these operating systems for deployment within the exacting and demanding automotive environment. Each system is scrutinized through various lenses, meticulously exploring their capacities, adaptabilities, and limitations within the intricate ecosystem of modern vehicle engineering. This comprehensive analysis endeavours to provide invaluable insights for automotive engineers, system designers, and stakeholders in the automotive industry. By elucidating the intricate nuances of these operating systems, the paper aims to guide informed decision-making processes regarding the selection and implementation of RTOS solutions, thereby advancing the realm of automotive technology towards higher levels of efficiency, reliability, and safety.

Keywords— Automotive embedded systems, real-time operating systems, fault-tolerant ecus, autosar architecture, engine control unit software.

I. INTRODUCTION

The integration of Engine Control Units (ECUs) into vehicles has marked a significant transformation in the automotive industry, facilitating precise engine performance optimization and control. The effectiveness of these ECUs relies intricately on the ability of the underlying operating system to meet strict real-time demands imposed by automotive applications. Real-time operating systems (RTOS) assume a pivotal role in ensuring deterministic precision in executing critical tasks such as sensor data processing, actuator control, and seamless communication with other vehicle systems.

This research paper aims to navigate the intricate landscape of operating systems embedded within Engine Control Units. The focal point of examination lies in RTOS solutions that play an instrumental role in preserving temporal integrity within automotive control systems. Specifically, an analysis will be conducted on Embedded Kernel, Vehicle Distributed Executive (OSEK/VDX), AUTOSAR, FreeRTOS, QNX, Integrity, and Linux with real-time patches. Each of these operating systems possesses distinctive features and trade-offs, directly impacting their applicability and feasibility within ECUs.

The overarching objective of this study is to furnish a comprehensive understanding of the strengths and limitations of diverse RTOS options in the context of automotive applications. By scrutinizing key attributes—real-time responsiveness, scalability, reliability, and integration ease—the aim is to equip automotive engineers, researchers, and developers with insights crucial for making informed decisions while selecting an operating system for Engine Control Units.

As we progress, subsequent sections of this paper will offer detailed analyses of each RTOS. These analyses will illuminate their architectural intricacies and performance characteristics, contextualizing their roles within real-time automotive scenarios. The exploration will intertwine the specific features of these operating systems with their practical implications within the automotive ecosystem, unravelling the nuances that underlie their functionality and relevance within ECUs.

II. THEORETICAL RESEARCH ON AVAILABLE EMBEDDED OPERATING SYSTEMS

2.1 Real-Time Operating Systems(RTOS)

RTOS serves as the cornerstone for time-critical applications within ECUs, ensuring precise task execution and deterministic behaviour. Tailored for safety-critical systems, RTOS focuses on efficient task and resource management, meeting stringent timing requirements in automotive applications such as engine control and driver-assistance systems. Its core lies in implementing scheduling algorithms like Rate Monotonic Scheduling (RMS) and Earliest Deadline First (EDF), guaranteeing timely task completion. RTOS offers key features including deterministic task scheduling, low interrupt latency,

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Volume no 11 Issue 1 January 2024

priority-based task management, memory protection, fault tolerance, and pre-emptive multitasking. These functionalities enable optimal resource utilization and responsiveness, crucial in modern vehicles' multitasking environments.

2.1.1 OSEK/VDX (Operating System Embedded Kernel/Vehicle Distributed eXecutive)

OSEK/VDX, a fundamental framework for embedded systems in vehicles, standardizes real-time operating systems (RTOS) to ensure reliable task scheduling and resource management. This framework prioritizes deterministic task execution, crucial for safety-critical functions within automotive Engine control units (ECUs). However, configuring these RTOS for specific ECUs often involves manual processes due to the absence of comprehensive automation tools.

To address these challenges, various approaches have emerged, including manual file generation, GUI-based tools offering guidance, visual designers like SmartOSEK for automatic file creation, and frameworks supporting model-based development. Mader's prototype toolchain complements this landscape by aiming to facilitate continuous safety analysis. It establishes a centralized repository for Information, fostering seamless collaboration across engineering domains and specialized tools.

The integration of this toolchain with Enterprise Architect and other specialized tools ensures consistency and traceability across different artifact types and domains. This represents a shift from document-centric approaches to more efficient, interconnected model-based development, aligning with ISO 26262 standards. OSEK/VDX's emphasis on deterministic task execution complements this effort, reinforcing the critical role of reliable task scheduling and resource management in automotive safety-critical functions.

Key Features:

- Task Scheduling Precision: OSEK/VDX ensures tasks adhere to predefined time constraints, vital for maintaining vehicle safety and reliability.
- Minimal Interrupt Latency: Reducing interrupt latency enables swift responses to real-time events, critical in applications like collision avoidance.
- Priority-Driven Task Management: Prioritizing tasks ensures immediate attention to essential functions, enhancing vehicle safety.
- Fault-Tolerant Mechanisms: Robust fault tolerance strategies prevent system failures and data corruption, enhancing ECU reliability.
- Pre-emptive Multitasking: Supporting multitasking allows for optimal resource use and responsiveness in complex automotive environments.

2.1.2 FreeRTOS

FreeRTOS, specifically designed for embedded systems, emphasizes real-time task scheduling and management within engine control units (ECUs). Its core objective revolves around ensuring the efficient execution of critical functions in diverse automotive applications. By functioning as a pre-emptive multitasking kernel, FreeRTOS enables the concurrent execution of tasks based on assigned priorities, catering to the real-time demands of these applications.

The system's adaptability across various microcontroller architectures and its ability to handle limited resources make it a suitable choice for ECUs in automobiles. Its small footprint and minimal overhead accommodate the stringent resource constraints often present in automotive embedded systems.

Additionally, FreeRTOS's open-source nature facilitates active community involvement, fostering continuous improvements and comprehensive support through forums and extensive documentation. Furthermore, its integration within the RISC-V ecosystem further broadens its applicability, enabling it to address the real-time demands of automotive systems built on the RISC-V architecture. This integration underscores FreeRTOS's role in delivering scalable and efficient embedded solutions tailored to the needs of automotive technology, catering to the critical functions and tasks within these systems.

Key Features:

- Deterministic Task Scheduling: FreeRTOS schedules tasks based on priorities, meeting stringent timing requirements in safety-critical systems.
- Low Interrupt Latency: Minimized latency allows quick responses to real-time events, crucial for time-sensitive automotive functions.
- Priority-Based Task Control: Prioritization ensures immediate handling of critical functions, bolstering vehicle safety measures.
- Memory Protection and Resilience: Incorporating robust memory protection and fault tolerance safeguards data and system reliability.
- Pre-emptive Multitasking: Simultaneous task execution optimizes resource utilization in multifaceted automotive environments.

2.1.3 QNX

QNX stands out for its reliability as a real-time operating system (RTOS), earning recognition in the industry for its robustness and scalability. It's specifically tailored for safety-critical applications, notably within automotive engine control units (ECUs). This operating system is engineered to seamlessly integrate and deliver high performance across a multitude of electronic systems within vehicles.

Acknowledged for its reliability, QNX has gained prominence as a trusted solution, particularly in environments where safety is paramount. Its robustness ensures the consistent and predictable execution of critical functions, essential for maintaining the safety and efficiency

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Volume no 11 Issue 1 January 2024

of automotive systems.

Furthermore, QNX's scalability allows it to adeptly adjust to the complexities inherent in modern electronic systems found in vehicles. It accommodates a diverse array of electronic components and functionalities, providing a stable and dependable platform for a wide spectrum of automotive applications.

Its capability to meet the exacting demands of safety-critical applications within automotive ECUs underscores its reputation for reliability, scalability, and seamless integration. QNX plays a pivotal role in ensuring the smooth and secure operation of electronic systems within vehicles, prioritizing safety and efficiency in the automotive sector.

Key Features:

- High Reliability and Fault Tolerance: QNX guarantees reliability, crucial for safety-critical functions in automotive ECUs, even amidst hardware or software failures.
- Real-Time Precision: Precise and predictable task execution is vital for critical automotive functions like braking and collision avoidance.
- Symmetric Multiprocessing (SMP) Support: SMP enhances processing power, allowing parallel task execution without compromising real-time capabilities.
- Robust Message Passing: Efficient inter-process communication ensures seamless integration of diverse automotive functionalities.
- Safety-Critical Application Support: Comprehensive support for critical automotive systems like adaptive cruise control and collision warning systems.

2.1.4 INTEGRITY

INTEGRITY, a distinguished real-time operating system (RTOS), stands out in effectively managing safety-critical applications within automotive engine control units (ECUs). It's revered for its high reliability, emphasizing robust partitioning and secure memory protection mechanisms crucial for ensuring dependable and secure vehicle operation.

Specifically tailored for safety-critical environments, INTEGRITY shines in its ability to deliver consistent and predictable execution of critical functions within automotive systems. Its core strengths lie in providing a dependable platform with robust partitioning, ensuring that different applications within the ECUs operate independently, preventing interference and enhancing system reliability.

Moreover, INTEGRITY's emphasis on secure memory protection mechanisms contributes significantly to the safety and security of vehicle operations. By securely isolating and protecting critical system components, it ensures a reliable and safe operational environment for automotive electronic systems.

INTEGRITY's excellence in managing safety-critical applications within automotive ECUs is attributed to its high reliability, robust partitioning mechanisms, and secure memory protection features. These qualities solidify its role as a dependable and secure real-time operating system, essential for maintaining the reliability and security standards in automotive technology.

Key Features:

- Reliability and Robustness: INTEGRITY prioritizes reliability, crucial for managing safety-critical applications in vehicle ECUs.
- Secure Partitioning: Isolation and protection of critical tasks and data ensure security and integrity within the ECU.
- Real-Time and Embedded Support: Comprehensive backing for real-time and embedded applications enhances overall ECU performance and responsiveness.
- Scalability and Adaptability: Flexible design accommodates varied automotive ECU architectures and evolving vehicle needs.
- Compliance with Safety Standards: Adherence to automotive safety standards like ISO 26262 ensures regulatory compliance and functional safety.

2.2 General-Purpose Operating Systems with Real-Time Capabilities

General-purpose operating systems with real-time capabilities provide a versatile platform accommodating various applications, including those requiring real-time responsiveness within ECUs. While not exclusively dedicated to real-time tasks like RTOS, these systems integrate features enabling efficient handling of time-sensitive functions. Leveraging scheduling algorithms such as Completely Fair Scheduler (CFS) and Deadline Scheduler, these OS variations support multitasking and prioritize tasks for optimal performance, especially critical in automotive environments. They offer adaptability, allowing customization and real-time patching to meet specific timing requirements in safety-critical applications. These OS options strike a balance between general computing tasks and the responsiveness needed for time-critical automotive functions.

2.2.1 Linux (with real-time patches)

Embedded Linux, augmented with real-time patches, emerges as a versatile platform suited to address the intricate demands of automotive Engine Control Units (ECUs). This amalgamation provides a robust foundation facilitating multitasking capabilities, accommodating diverse communication protocols, and offering customizable solutions within vehicular engine systems.

In the automotive domain, the integration of Embedded Linux empowered by real-time patches creates a flexible framework capable of simultaneous task execution, ensuring timely responses to critical events in engine control. Real-time patches augment Embedded Linux, enabling it to proficiently manage essential real-time functions while accommodating a spectrum of applications within Engine Control Units.

Furthermore, the adaptability of Embedded Linux, strengthened by real-time capabilities, facilitates the integration of various communication protocols crucial for automotive engine applications. This adaptability fosters seamless connectivity and communication among different engine systems, enhancing overall functionality and interoperability.

The amalgamation of Embedded Linux with real-time patches represents a significant solution for Engine Control Units, offering adaptability, extensive functionality, multitasking capabilities, support for diverse communication protocols, and customizable features. This combination serves as a compelling and robust platform, well-suited for meeting the dynamic requirements of contemporary vehicular engine systems.

Key Features:

- Open-Source Flexibility: The Customizable nature allows tailoring to specific ECU requirements, fostering innovation and unique functionalities.
- Community Support and Development Tools: Rich ecosystem facilitates collaboration and continuous improvement in Embedded Linux for automotive applications.
- Multitasking Capabilities: Supports simultaneous execution of tasks, enhancing user experience and functionality within the vehicle.
- Diverse File Systems and Networking: Compatibility with various file systems and networking protocols ensures seamless integration and connectivity.
- Real-Time Patch Options: Real-time patches address timing requirements for critical applications, ensuring reliability in demanding automotive environments.

2.2.2 AUTOSAR (AUTomotive Open System ARchitecture)

It is standardized software architecture for Engine Control Units (ECUs), known as AUTOSAR (AUTomotive Open System ARchitecture), comprises two platforms: the Classic Platform (CP) and the Adaptive Platform. While the CP caters to deeply embedded low-complexity devices, the Adaptive Platform addresses the evolving needs of modern automotive systems, particularly in high-performance computing and communication.

The Adaptive Platform extends its support to many core processors, heterogeneous computing platforms, and high-bandwidth communication technologies like Ethernet. This enables parallel processing and offers flexibility through Service-Oriented Architecture (SOA), allowing dynamic service and client linking during runtime, providing developers an advantage.

Moreover, the AUTOSAR Adaptive Platform relies on Scalable service-Oriented MiddlewarE over IP (SOME/IP) to ensure scalability across devices of various sizes and operating systems, including small devices like cameras, telematics devices, and infotainment devices.

In the domain of Engine Control Units, AUTOSAR provides standardized software frameworks. These frameworks facilitate the integration of diverse functionalities and allow flexibility in software configuration. AUTOSAR also supports essential features for dependable engine control systems, such as safety and security functionalities like priority-based scheduling, authenticated code execution, and controlled memory and CPU resource allocation.

Key Features:

- Standardized Software Architecture: Defines software structure for seamless integration and collaboration among various automotive systems.
- Modularity and Reusability: Modular design supports the development of reusable software components, fostering innovation and collaboration.
- Scalability for ECU Platforms: Adaptable to diverse ECU requirements, ensuring compatibility and optimal performance within vehicle systems.
- Interoperability across Systems: Facilitates communication and integration among different ECUs and electronic systems, enhancing vehicle connectivity.
- Support for Advanced Driver-Assistance: Enables the development of sophisticated ADAS features for enhanced vehicle safety and convenience.

III. FINAL DISCUSSION

The diverse array of operating systems integrated within Engine Control Units (ECUs) reveals a spectrum of functionalities and trade-offs that significantly influence the automotive industry's evolution. Real-Time Operating Systems (RTOS) occupy a central role within automotive control systems, ensuring the deterministic execution and responsiveness of critical tasks.

A thorough analysis encompassed Embedded Kernel, Vehicle Distributed Executive (OSEK/VDX), AUTOSAR, FreeRTOS, QNX, Integrity, and Linux with real-time patches. Each RTOS exhibits unique attributes, encapsulating varying degrees of real-time responsiveness, scalability, reliability, and integration ease, shaping their applicability within ECUs.

Fundamental frameworks like Embedded Kernel and OSEK/VDX prioritize deterministic task execution essential for safety-critical functions but encounter challenges in manual configuration. AUTOSAR's standardized architecture, addressing Classic and Adaptive Platforms, emphasizes modularity and scalability, accommodating diverse ECU requirements and communication protocols.

FreeRTOS stands out for adaptability across microcontroller architectures, suitable for resource-constrained automotive systems. QNX's robustness and scalability establish it as a trusted solution, ensuring reliability and seamless integration across multiple electronic systems in vehicles.

INTEGRITY's emphasis on secure partitioning and memory protection ensures the management of safety-critical applications, guaranteeing reliable and secure operations. Linux, enhanced with real-time patches, presents an open-source, customizable platform capable of multitasking and supporting various communication protocols, offering flexibility in engine control applications.

This comprehensive analysis of RTOS within automotive applications highlights a mosaic of strengths and limitations, crucial for informed decision-making among automotive engineers and developers. Evaluating real-time responsiveness, scalability, reliability, and integration ease becomes imperative when selecting an operating system tailored to modern vehicle technologies.

This exploration underscores the intricate relationship between operating systems and the effectiveness of Engine Control Units, emphasizing the importance of balancing real-time demands with scalability, reliability, and integration ease. By illuminating the nuanced attributes of each RTOS, this study aims to serve as guidance, empowering stakeholders in the automotive industry to navigate the complex landscape of operating systems, and facilitating informed decisions crucial for advancing automotive control systems.

REFERENCES

[1] Macher, G., Atas, M., Armengaud, E., & Kreiner, C. (2015). Automotive real-time operating systems. ACM SIGBED Review, 11(4), 67–72.

[2] Lingga, W., Budiman, B. A., & Sambegoro, P. (2019). Automotive Real-Time Operating System in Vehicular Technology Progress Review. 2019 6th International Conference on Electric Vehicular Technology (ICEVT).

[3] Mendonca, L. S., Luceiro, D. D., Martins, M. E. S., & Bisogno, F. E. (2017). Development of an engine control unit: Implementation of the architecture of tasks. 2017 IEEE International Conference on Industrial Technology (ICIT).

[4] Cuatto, T., Passeronge, C., Lavagno, L., Jurecska, A., Damiano, A., Sansoè, C., … Sangiovanni-Vincentelli, A. (1998). A case study in embedded system design. Proceedings of the 35th Annual Conference on Design Automation Conference - DAC '98.

[5] Bhat, A., Samii, S., & Rajkumar, R. R. (2020). Fault-Tolerance Support for Adaptive AUTOSAR Platforms using SOME/IP. 2020 IEEE 26th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA).

[6] A thesis submitted for the degree of Master of Science 2nd September 2011 Supervised by Prof. Jim Woodcock and Dr. Matthew Naylor on FreeRTOS and Multicore.

[7] Steffen Kollman, Victor Pollex, Kilian Kempf, Frank Slomka, Matthias Traub, et al.. Comparative Application of Real-Time Verification Methods to an Automotive Architecture. 18th International Conference on Real-Time and Network Systems, Nov 2010, Toulouse, France. pp.89-98.

[8] Carlos Garre, Domenico Mundo, Marco Gubitosa, Alessandro Toso, "Real-Time and Real-Fast Performance of General-Purpose and Real-Time Operating Systems in Multithreaded Physical Simulation of Complex Mechanical Systems", Mathematical Problems in Engineering, vol. 2014, Article ID 945850, 14 pages, 2014. https://doi.org/10.1155/2014/945850.

[9] Georg Macher, Eric Armengaud, and Christian Kreiner. Automated Generation of AUTOSAR Description File for Safety-Critical Software Architectures. In Lecture Notes in Informatics, 2014.

[10] Zou, Y.; Zhang, W.; Weng, W.; Meng, Z. Multi-Vehicle Tracking via Real-Time Detection Probes and a Markov Decision Process Policy. Sensors 2019, 19, 1309.

[11] Wenying Zuo, Yinguo Li, Fengjuan Wang, Xiaobo Hou. "A New Design Method of Automotive Electronic Real-time Control System." Chongqing University of Posts and Telecommunications. 2012.

[12] OSEK/VDX: OSEK/VDX Operating System Specification Version 2.2.2., July 5 (2004).