Journal of Technological Innovations

Est. 2020

Hardware Prototyping using FreeRTOS while developing AutoSAR compliant application software using Simulink™

Roopak Ingole[1], Benjamin Eckhart[2]
Email: roopak.ingole@cummins.com[1], benjamin.eckhart@cummins.com[2]

Abstract:

This paper delves into an innovative approach to hardware prototyping for automotive software development within the AutoSAR framework, utilizing FreeRTOS in conjunction with Simulink. The automotive industry's shift towards software-centric systems necessitates reliable and efficient development practices. AutoSAR provides a standardized architecture that simplifies this process, while Simulink enhances rapid prototyping capabilities through its model- based design environment. FreeRTOS offers a lightweight, adaptable solution for real-time operating needs in embedded systems, supporting a broad spectrum of microcontroller architectures. By integrating FreeRTOS with AutoSAR and Simulink, this research presents a methodology that accelerates the development cycle, ensuring early validation of hardware and software components. The paper highlights the seamless synergy between these technologies, outlining their practical applications and the substantial benefits they offer in terms of development speed, software quality, and cost efficiency. The findings suggest that this integrated approach not only streamlines the prototyping process but also sets the stage for further innovations in automotive technology development. Through this exploration, we establish a robust framework for rapid prototyping, demonstrating significant advancements in the evaluation and implementation of new automotive features, thus contributing to the ongoing evolution of the automotive industry.

1.INTRODUCTION

The automotive industry is currently undergoing a significant transformation, with software becoming an integral part of automotive development. AutoSAR (**AUT**omotive **O**pen **S**ystem **AR**chitecture) [1] plays a pivotal role in this evolution, providing a standardized framework that facilitates the development of automotive software with high levels of reliability and efficiency. Simulink, a MATLAB-based graphical programming environment, further enhances this process by offering a platform for model-based design, which is instrumental in rapid prototyping. In the drive towards electrification, stable electronic hardware is key. This paper explores the method of Hardware prototyping using FreeRTOS

while leveraging synergy between AutoSAR and Simulink in the context of rapid prototyping, emphasizing the benefits, methodology, and practical applications of integrating these technologies.

2. ELECTRONIC HARDWARE DEVELOPMENT

For any new product, electronic hardware development starts with deep research for an approximate microcontroller. Microcontroller holds the brain of the product. It becomes crucial to verify all the aspects of the microcontroller during initial phases of hardware development.With the growing complexity of microcontrollers, push from simple compute engines towards System on Chip (SoC) drives the need for software platform capable of a higher level of validation. Along with SoC, new products, especially for electrified components for Automotive, require specialized integrated circuits (ICs) [2], ex. Battery Management Systems ICs. Most of the automotive chip manufacturers provide AutoSAR compliant microcontroller abstraction layer (MCAL) [3] and/or complex device drivers (CDD) that can be used for faster board powerup and verification of these functionalities. However, full AutoSAR workflow requires toolchain and expertise

explained in detail below.

Because of this reason, companies need more efficient prototyping solutions, especially during the hardware selection phase, so that the team can finalize the hardware without worrying about the complexity of the full AutoSAR stack and toolset. This is the area where we found FreeRTOS shines.

A. FreeRTOS

FreeRTOS [4] is an open-source real-time operating system (RTOS) designed for embedded systems. This provides a small, efficient kernel designed for microcontrollers and small microprocessors. It allows developers to create embedded applications with real-time requirements. This RTOS satisfies all the critical functionalities required for hard real-time applications. This is highly portable and supports a wide range of microcontroller architectures and development environments. This RTOS can be easily adapted to various hardware platforms. Most of the microcontroller manufacturers provide the controller specific base software package specific to the microcontroller. FreeRTOS has a large and active community of developers and users. Maintainers of this RTOS also provide middleware components and libraries for additional functionalities needed for embedded product development. Overall, FreeRTOS offers a robust, lightweight, and highly flexible solution for developing real-time embedded systems, with features tailored to meet the requirements of a wide range of critical applications

B. The AutoSAR Standard: Foundation for Automotive Software Development

AutoSAR [1] [3] [5] is a global development partnership of automotive interested parties founded with the aim of creating and establishing open standards for automotive E/E (Electrical/Electronic) architectures. This initiative enables the development of reusable and transferable software and hardware components, facilitating scalability and flexibility in automotive systems design. The standard encompasses a wide range of automotive applications, from simple control tasks to highly complex and safety-critical systems. By abstracting hardware and software, AutoSAR allows for the separation of application development from underlying hardware, a crucial feature required for rapid prototyping. Typical AutoSAR framework is as shown below: [6] Fig. 1. AutoSAR Layered Architecture.

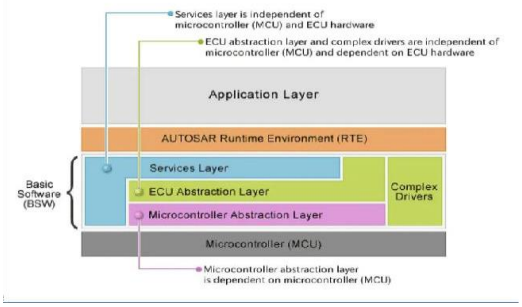


Fig. 1. AutoSAR Layered Architecture

In this layered architecture, AutoSAR prescribes the standardized interfaces between each layer. This allows the development of reusable components across the layers of the system. AutoSAR also prescribes the standardized methodology for designing and configuring the layers of the software and standardized communication protocols between ECUs.

Following the standardization, each architecture layer supplier/developer provides the software stack and configuration tool associated with it. Microcontroller manufacturers, like Infineon, NXP, Texas Instruments, Renesas, Analog Devices, provide the microcontroller abstraction layer (MCAL). Middleware Base software (BSW) layer developers like Vector, ETAS, KPIT [7] provides software stack for all of these services, RTOS, standard device drivers and communication stack along with configuration tools. For the application layer development, developers can use tools like Vector DaVinci Developer [8], ETAS ISOLAR A [9], or MathWorks's Simulink AutoSAR Composer [10]. Once all the tools are configured, tools like Vector DaVinci Configurator [11] or ETAS ISOLAR B [9] are used to connect everything and generate the code to compile and produce the executable.

2. RAPID PROTOTYPING WITH SIMULINK IN AN AUTOSAR CONTEXT

Simulink's model-based design [12] approach is ideally suited for rapid prototyping in an AutoSAR framework.

Simulink AUTOSAR Composer [10] is a powerful tool designed to streamline the development of automotive software according to the AUTOSAR standard. This software solution, developed by The MathWorks, enables automotive engineers to efficiently design, simulate, and implement AUTOSAR-compliant software architectures directly within the Simulink environment. With Simulink AUTOSAR Composer, users can model software components, interfaces, and behavior using a graphical interface, facilitating rapid prototyping and iteration. Moreover, it provides comprehensive support for AUTOSAR XML descriptions, ensuring compatibility and compliance with industry standards. By integrating seamlessly with other The MathWorks tools like Embedded Coder, Simulink AUTOSAR Composer enables automatic code generation for various target platforms, expediting the deployment of embedded automotive applications. Overall, Simulink AUTOSAR Composer empowers automotive developers to accelerate the development lifecycle, improve software quality, and achieve greater efficiency in the creation of AUTOSAR-compliant software architectures. This process not only accelerates the development cycle but also enhances the reliability of the software by enabling early detection and correction of design errors. Furthermore, Simulink's integration with AutoSAR facilitates the simulation and testing of automotive software within a virtual environment, allowing for extensive validation before deployment on actual hardware.

4. CHALLENGES WITH AUTOSAR TOOLS IN THE CONTEXT OF RAPID HARDWARE PROTOTYPING

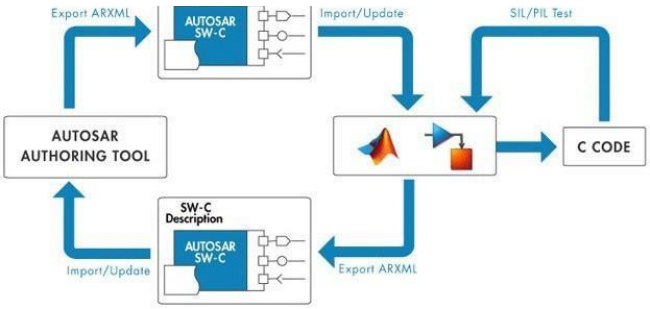
As described above, AutoSAR is an excellent methodology for developing embedded automotive software; however, it provides its own overheads and challenges while adapting to early hardware development. Utilizing a full AutoSAR stack requires all tools to be configured for the chosen microcontroller sufficiently early to produce the software. During the hardware development phase, hardware engineers must receive early feedback on their functionality and meet the requirements for the proposed application. In parallel controls, application software teams would like to start developing a control application geared towards production functionality. This poses a chicken and egg problem. If hardware and software are developed sequentially, this will result in significant delays in product development and loss of critical time. AutoSAR provides significant advantages for production-intent software development and provides the necessary flexibility for prototyping existing products. However, for brand-new products with brand-new microcontrollers, the process usually starts with the MCAL layer being available from the microcontroller manufacturer, and then BSW providers will start configuring their middleware stack for a given microcontroller. This usually takes months of effort before the fully configured MCAL and BSW are provided to end customers. While all of this configuration of the MCAL and BSW was ongoing, the controls application team needed to know the interfaces required to interact with the hardware. This delays the overall hardware and software development process. Along with this large investment in AutoSAR toolset licenses is required. To solve this problem, reduce the cost, and reduce the dependency between hardware and software, I decided to explore open-source FreeRTOS for early hardware prototyping while allowing the application team to develop an AutoSAR application layer.

5. METHODOLOGY FOR INTEGRATING AUTOSAR/SIMULINK WITH FREERTOS

Engaging in the integration of AutoSAR and Simulink, our methodology embarks with an initial stage focused on outlining the system's architectural design and pinpointing essential components, as delineated by AutoSAR guidelines. This phase is critical for laying the foundational elements of our project. Moving forward, we harness the capabilities of Simulink to craft and test simulations of these components, aiming for unparalleled performance. The next step involves transforming the Simulink models into code that aligns with AutoSAR standards through the use of sophisticated automatic code generation utilities. This AutoSAR-ready code is then seamlessly woven into the AutoSAR Runtime Environment (RTE), setting the stage for exhaustive testing and validation. Emblematic of a truly iterative process, our approach is characterized by recurrent feedback mechanisms, ensuring the system's design is continually refined and optimized.

In the context of pioneering a new Battery Management System (BMS) Integrated Circuit (IC) for our forthcoming battery product series, our team embarked on an evaluative journey to discern the most suitable microcontroller and BMIC, alongside the optimal configuration for our BMS's electrical and electronic architecture. This endeavor necessitated the creation of a swift hardware evaluation platform coupled with the development of an AutoSAR-compliant application layer. Our goal was to facilitate the effortless deployment of applications across diverse microcontroller environments, thereby enhancing our project's adaptability and efficiency.

AutoSAR suggested a Top-Down or Bottom-Up approach Fig. 2. Workflows for AUTOSAR [13], we took both the approaches at the same time. The process we developed for BMS hardware and software development is as shown in Fig.
3. Autosar application in Simulink integrated with FreeRTOS



device drivers, and a communication stack. To align closely with our microcontroller's architecture, we opted for the FreeRTOS port, with a specific focus on the version tailored for Infineon's TC27x series [16], which served as our initial benchmark. This choice was strategic, considering our ultimate target: the Infineon TC377 microcontroller. Upon selecting this platform, we meticulously configured the RTOS to suit the intricacies of the TC377, paying special attention to its multi-core capabilities.

Fig. 2. Workflows for AUTOSAR

A. Application Software Development

In the pursuit of crafting sophisticated application software, our team turned to the Simulink AutoSAR Composer [14] [15]. This powerful tool enabled us to design a comprehensive application layer, complete with all necessary components and their configurations. We meticulously implemented interfaces among these components and seamlessly integrated a AutoSAR dictionary within the Simulink framework. AutoSAR dictionary provided a user interface to capture AutoSAR specific information e.g. ports, datatypes, calibration, and internal behavior. Leveraging the simulation capabilities of Simulink, we conducted thorough verifications of each component's functionality and their interconnections

Further enhancing our development process, we incorporated a plant model crafted in Simscape. This inclusion facilitated close-loop simulations, allowing us to rigorously test and confirm the effectiveness of our control strategies. Upon successful validation, the Simulink Coder feature was employed to translate our models into AutoSAR-compliant C code, generate component and composition ARXMLs, and produce A2L artifacts for each component.

To orchestrate the management of this complex application project, our strategy encompassed the utilization of Simulink Project and GitLab. This approach not only streamlined code versioning but also fostered collaboration among our developers, ensuring a cohesive and efficient development process. The resultant C code emerged as a pivotal element, primed for integration with the RTE and BSW layers of AutoSAR.

Our collaboration with The MathWorks was instrumental in this endeavor, particularly in the implementation and demonstration of multi-instance software component capabilities within Simulink. This feature proved indispensable for applications such as the Battery Management System, which requires the execution of identical calculations across multiple cells within a battery pack, showcasing our commitment to innovation and precision in software development for advanced systems.

B. Base Software Development

Our base software architecture encompasses a real-time operating system (RTOS), middleware components, Given the TC377's multi-core architecture, our strategy involved deploying the RTOS exclusively on Core0, while opting to manage the remaining cores using a bare metal approach. This decision allowed us to tailor the system's performance and resource allocation more precisely. To facilitate communication with Battery Management System Integrated Circuits (BMICs) and host computers, we crafted bespoke device drivers for CAN, SPI, UART, and Ethernet interfaces.

Our integration efforts extended to incorporating a lightweight IP stack, leveraging Vector's open-source XCP stack [17], and implementing a simple shell via the UART interface. For inter-core communication, we utilized straightforward, memory-based techniques, ensuring efficient data transfer between cores. This foundational software layer was developed concurrently with the application software, ensuring a harmonious integration process.

With the base software infrastructure firmly in place, complete with fully functional interfaces for device drivers and the RTOS, our next course of action revolves around bridging the Base Software (BSW) and Application Software (ASW) through the Real-Time Environment (RTE). This integration is pivotal, as it promises to enhance the system's overall functionality and performance, laying a solid groundwork for advanced operational capabilities.

C. RTE Generation

In our development process, instead of employing

formal AutoSAR tools for configuring the Base Software (BSW) and automatically generating the Real-Time Environment (RTE), we took a different approach. We capitalized on the auto- generated RTE stub functions provided by Simulink Coder. These stub functions served as the foundation upon which we manually coded the necessary connections, effectively bridging the BSW with the application layer.

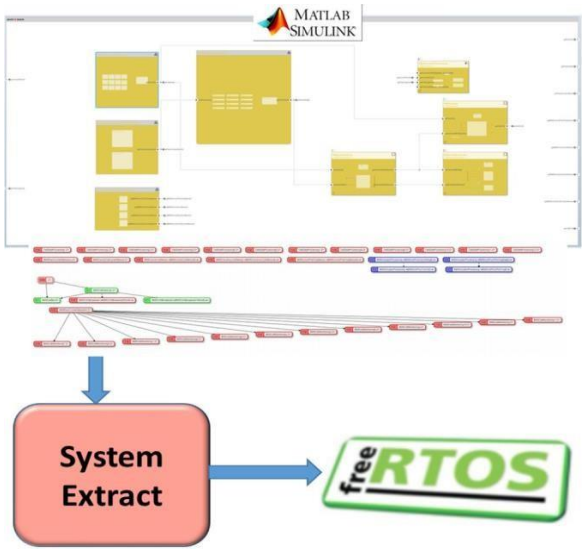
The decision to hand-code these stub functions was strategic, driven by the standardized interfaces of our base software. These standardized interfaces significantly streamlined the realization of the stub functions, allowing for a more direct and efficient integration process. This approach not only facilitated a smoother melding of the BSW with the application layer but also underscored our flexibility and adaptability in software development, ensuring that even without the direct use of formal AutoSAR tools, we could achieve a seamless and robust integration.

D. Compilation

For the compilation phase, we harnessed the capabilities of the TASKING TriCore Integrated Development Environment (IDE) [18], which served as the central hub for amalgamating the project's components: RTOS, BSW, RTE, and Application Software (ASW). Within the TASKING

TriCore IDE, we integrated the automatically generated ASW code, manually coded RTE, custom middleware, and configured both the Microcontroller Abstraction Layer (MCAL) [19] and RTOS into a singular project, executing a comprehensive compilation of the code.

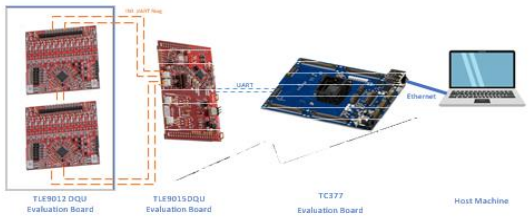
Upon successful compilation, which culminated in the creation of the final executable files (.elf and .map), our next endeavor was to generate A2L files. This file is essential for calibration tool purposes, facilitating the precise mapping of all variables to their respective memory locations through CANape's A2L mapping functionality [20]. The culmination of this meticulous process resulted in two key outputs: the executable file (.elf) and the comprehensive A2L file (.a2l), marking a significant milestone in our project's development. Whole integration workflow is depicted in Fig. 3. Autosar application in Simulink integrated with FreeRTOS.



E. . *Verification & Validation*

Prior to the deployment on our tailor-made hardware platform, it was imperative to validate the capabilities of both the microcontroller and the Battery Management System Integrated Circuit (BMIC) using the evaluation boards provided by the chip manufacturer. For the Infineon TC377 microcontroller, the KIT_A2G_TC377 [21] evaluation board was employed, while the TLE9012DQU Evaluation Board [22] and TLE9015DQU Evaluation Board [23] were utilized for the Infineon's BMIC [2]. These evaluation kits, coupled with minimal electronic assembly, swiftly set the stage for our verification platform Fig. 4. Hardware Evaluation Test Setup.

Employing Infineon’s Memtool [24], we proceeded to flash the compiled binary onto the target microcontroller. This step was crucial for initiating the verification process. Using a UART console and Vector’s Canape, we were able to meticulously verify all the requisite functionalities of both the BMICs and microcontrollers. This process allowed us to assess the BMIC communication efficiency and the robustness of the multi-core processing capabilitiesThe verification phase, conducted with these tools and platforms, provided us with a comprehensive understanding of the system's performance, ensuring that all components functioned in harmony and met our stringent requirements. This meticulous approach to validation was instrumental in paving the way for the successful implementation of our custom-designed hardware platform, ensuring a high degree of reliability and performance in real-world applications.



6. RESULTS

Embarking on this development journey enabled us to establish a robust framework for both hardware and software prototyping, marking a significant milestone in our engineering endeavors. A pivotal aspect of our success was the validation of a system comprising eight Battery Management System Integrated Circuits (BMICs) interconnected via ISO UART in a ring topology. To emulate the cells, we used Battery Cell emulators along with real coin cells. This venture, characterized as new, unique, and difficult (NUD), posed a considerable challenge, especially given the ambitious timeframe for validation.

Furthermore, our validation efforts extended to the verification of ISO SPI connectivity, the exploitation of multi- core processing capabilities, and the implementation of crucial cell-specific algorithms designed to manage up to 180 cells per battery pack. This comprehensive approach culminated in the development of a next-generation Battery Management System (BMS) platform, fully equipped with both the hardware and software specifications required to propel our projects forward.

This streamlined process not only facilitated the rapid verification of all critical NUDs but also enabled the efficient derivation of requirements within an unprecedented timeframe. Under conventional circumstances, utilizing a standard AutoSAR-based development approach might have extended the verification phase to as long as 18 months. However, thanks to our innovative methodology, we managed to significantly expedite this process, achieving our objectives within a mere 6-8 months.

A key factor in the acceleration and cost-efficiency of our development process was the strategic utilization of new open-source tools alongside existing resources from the Cummins software library. By leveraging FreeRTOS for our real-time operating system needs—a platform available at no cost for evaluation purposes—we avoided the financial and temporal expenditures associated with formal AutoSAR tooling. Moreover, our enterprise licensing agreement with The MathWorks facilitated unrestricted access to MATLAB/Simulink for a broad spectrum of electronic control system development tasks, without incurring additional licensing fees. This strategic decision negated the necessity for expensive AutoSAR tools such as Vector's DaVinci Developer and Configurator or ETAS ISOLAR A/B, yielding substantial cost savings during the crucial early phases of development. Thus, our innovative approach not only streamlined the validation process but also enabled informed decision-making regarding the selection of microcontrollers, BMICs, and the production toolchain, setting a new benchmark for efficiency and effectiveness in our development practices.

7. PRACTICAL APPLICATIONS, BENEFITS AND FUTURE ENHANCEMENT

The synergy of AutoSAR with Simulink for rapid prototyping unveils a horizon of possibilities, extending from the sophistication of hydrogen Fuel-Cells, Advanced Driver Assistance Systems (ADAS), Autonomous Driving System (ADS) and the intricacies of powertrain control to the dynamic realms of infotainment and telematics. This methodology of swift hardware and software prototyping is a game-changer, significantly curtailing the time-to-market for introducing novel automotive features while upholding exemplary standards of software quality and adherence to industry benchmarks. Furthermore, the capability to emulate and scrutinize automotive software within a virtual milieu drastically diminishes development expenditures and circumvents the reliance on costly prototypes.

In the exploratory stages of groundbreaking technology or throughout the nascent phases of new product development, this streamlined process offers vital time and cost savings for organizations. It propels the acceleration of the incubation period for new products, thereby fostering innovation and rapid market entry. Importantly, this approach is not confined to Battery Management Systems (BMS) but is equally applicable across a spectrum of electronic hardware development endeavors. The advent of automation and a modicum of standardization in the processes of RTE generation and compilation could further refine this methodology. By minimizing manual interventions, these enhancements have the potential to significantly accelerate the innovation trajectory for new products, heralding a new era of efficiency and creativity in automotive development.

8. CONCLUSION

The culmination of our exploration into the synergistic integration of AutoSAR and Simulink underscores a transformative approach to rapid prototyping within the automotive sector. This fusion of AUTOSAR's standardized framework with Simulink's robust model-based design capabilities empowers engineers to streamline the development cycle. This process ensures the delivery of software that not only adheres to but also surpasses industry

benchmarks for quality. The strategic incorporation of FreeRTOS as the middleware for the Base Software (BSW) has markedly expedited the evaluation phase of technological ventures, significantly benefiting the incubation period of novel product development.

This collaboration between AutoSAR and Simulink extends beyond mere efficiency and quality enhancement; it serves as a beacon for innovation and technological progression in an industry characterized by increasingly complex systems. As automotive technologies evolve, the integrated approach offered by AutoSAR, Simulink, and FreeRTOS will become increasingly indispensable. These tools collectively forge a path towards the future, equipping developers with the means to navigate the complexities of automotive system design with greater agility and foresight. In essence, the integration of these platforms not only meets the current demands of automotive software development but also anticipates the future needs of an industry at the forefront of technological innovation.

9. REFERENCES

[1] AUTOSAR, "AUTOSAR," [Online]. Available: https://www.autosar.org/.

[2] Infineon Technologies AG, "Battery management ICs | Infineon's battery management ICs offer an optimized solution for cell monitoring and balancing," [Online]. Available: https://www.infineon.com/cms/en/product/battery-management-ics/.

[3] AUTOSAR, "AutoSAR Classic Platform - MCAL Layer," [Online]. Available: https://www.autosar.org/search?tx_solr%5Bfilter%5D%5B0%5D=c ategory%3AR22-11&tx_solr%5Bfilter%5D%5B1%5D=platform%3ACP&tx_solr%5 Bfilter%5D%5B3%5D=arhitectureElement%3AMicrocontroller+Dri vers&tx_solr%5Bfilter%5D%5B4%5D=workingGroup%3AWG- CP-MCL&tx_solr%5Bq%5D.

[4] Amazon Web Services, Inc., "The FreeRTOS™ Kernel," [Online]. Available: https://www.freertos.org/RTOS.html.

[5] AUTOSAR, "Autosar Classic Platform," [Online]. Available: https://www.autosar.org/standards/classic-platform.

[6] A. Abdelhakeem, "Classic Autosar - Quick Review. STACKS OVERVIEW | by abdullah abdelhakeem," [Online]. Available: https://medium.com/@AbdullahAbdelhakeem22/classic-autosar- quick-review-d10ef2a13678.

[7] KPIT Technologies, "Autosar Solutions," [Online]. Available: https://www.kpit.com/solutions/autosar/.

[8] Vector Informatik GmbH, "DaVinci Developer Classic," [Online]. Available: https://www.vector.com/int/en/products/products-a-z/software/davinci-developer-classic/#.

[9] ETAS Inc., "ISOLAR," [Online]. Available: https://www.etas.com/en/products/isolar.php.

[10] The Mathworks, Inc., "System Composer - MATLAB," [Online]. Available: https://www.mathworks.com/products/system-composer.html.

[11] Vector Informatik GmbH, "DaVinci Configurator Classic," [Online]. Available: https://www.vector.com/int/en/products/products-a-z/software/davinci-configurator-classic/.

[12] The Mathworks, Inc., "Simulink - Simulation and Model-Based Design - MATLAB," [Online]. Available: https://www.mathworks.com/products/simulink.html .

[13] D. Dandotiya, "Software Architecture & AUTOSAR for Automotive Embedded system," [Online]. Available: https://www.pathpartnertech.com/software-architecture-autosar-for- automotive-embedded-system/.

[14] The Mathworks, Inc., "AUTOSAR Blockset - MATLAB," [Online]. Available: https://www.mathworks.com/products/autosar.html.

[15] The Mathworks, Inc., "Workflows for AUTOSAR," [Online]. Available: https://www.mathworks.com/help/autosar/ug/workflow-for-autosar.html.

[16] A. Tengg, "FreeRTOS 7.1 Port for Aurix (TC27x), using Free Entry Toolchain," [Online]. Available: https://interactive.freertos.org/hc/en-us/community/posts/210026366-FreeRTOS-7-1-Port-for-Aurix- TC27x-using-Free-Entry-Toolchain?_ga=2.118106369.926239537.1710598264-626888096.1709996142.

[17] Vector Informatik GmbH, "XCP Sample Implementation," [Online]. Available: https://support.vector.com/kb?id=kb_article_view&sysparm_article=KB0011316&sys_kb_id=6351582c87706110cd36fd99cebb35e9&spa=1.

[18] TASKING, "TRICORE VX SOFTWARE DEVELOPMENTTOOLS," [Online]. Available: https://www.tasking.com/taxonomy/term/223.

[19] TASKING, "HOW TO BUILD YOUR ILLD APPLICATION WITH TASKING VX-TOOLSET FOR TRICORE," [Online]. Available: https://resources.tasking.com/sites/default/files/2021-02/How%20to%20build%20your%20iLLD%20application%20with%20TASKING%20VX-toolset%20for%20TriCore_WEB.pdf.

[20] Vector Informatik GmbH, "How do I create an A2L file fromCANape?," [Online]. Available: https://cdn.vector.com/cms/content/know-how/_application- notes/AN-IMC-1-024_How_do_I_create_an_A2L_file_from_CANape.pdf.

[21] Infineon Technologies AG, "KIT_A2G_TC377_5V_TRB_S,"[Online]. Available: https://www.infineon.com/cms/en/product/evaluation- boards/kit_a2g_tc377_5v_trb_s/.

[22] Infineon Technologies AG, "TLE9012DQU_DTR_BMS2,"[Online]. Available: https://www.infineon.com/cms/en/product/evaluation- boards/tle9012dqu_dtr_bms2/.

[23] Infineon Technologies AG, "TLE9015DQU_TRX_BRG," [Online].Available: https://www.infineon.com/cms/en/product/evaluation-boards/tle9015dqu_trx_brg/.

[24] Infineon Technologies AG, "Infineon Free Tools," [Online]. Available: https://www.infineon.com/cms/en/tools/aurix-tools/free-tools/infineon/.