Technical White Paper

MCAL and AUTOSAR for MSP Microcontrollers

Ethan Bloom

ABSTRACT

As automotive embedded systems increase in complexity and demand greater portability, having a standardized software architecture across ECUs from various suppliers has become essential. This white paper details how Texas Instruments' MCAL package for MSP Microcontrollers enables AUTOSAR® compliance across the MSPM0 and MSPM33 microcontroller families. By providing a standardized abstraction of core microcontroller functions, TI's MSP-MCAL package allows developers to decouple application logic from hardware-specific implementation, facilitating seamless integration with AUTOSAR's layered software components, reducing development complexity and accelerating time-to-market. Ultimately, TI's MSP-MCAL package provides a scalable, standards-based foundation for modern automotive embedded systems.

Table of Contents

Trademarks

1 Introduction

Increasing complexity in automotive applications is driving a heightened demand for software portability and functional safety. To address these requirements, the automotive industry has embraced standardized abstraction layers to bridge the gap between high-level application code and low-level hardware drivers. One of the industry-standard solutions to this problem is AUTOSAR, which allows for scalable hardware independence and seamless software integration. This document presents Texas Instruments' standardized MCAL package offering, and examines the deployment of AUTOSAR-compliant software on Texas Instruments' MSPM0 and MSPM33 microcontrollers. The discussion in this paper is organized into four primary areas:

- **Foundations**: An overview of the AUTOSAR architecture and the critical role of the Microcontroller Abstraction Layer (MCAL)
- **TI's MSP-MCAL Offering**: A deep dive into the features, safety compliance, and licensing model provided by TI's specialized MCAL package for MSP
- **Ecosystem Integration**: An analysis of how MSP-MCAL and other tools in the MSP software ecosystem scale across the MSP microcontroller portfolio
- **Getting Started**: High-level guidance for getting started quickly with MSP-MCAL and achieving successful software integration and deployment

2 MCAL and AUTOSAR Basics

Modern vehicles integrate hundreds of electronic control units (ECUs) spanning various vehicle systems and domains, such as body electronics, ADAS, and lighting. Each system has different levels of security and safety requirements, incorporated across ECUs from multiple vendors. Managing this complexity demands standardized software frameworks that enable portability, interoperability, and efficient development cycles. This need is addressed by an industry-standard layered framework called *AUTOSAR* (AUTomotive Open System ARchitecture), and its lowest-level component, the *Microcontroller Abstraction Layer (MCAL)*.

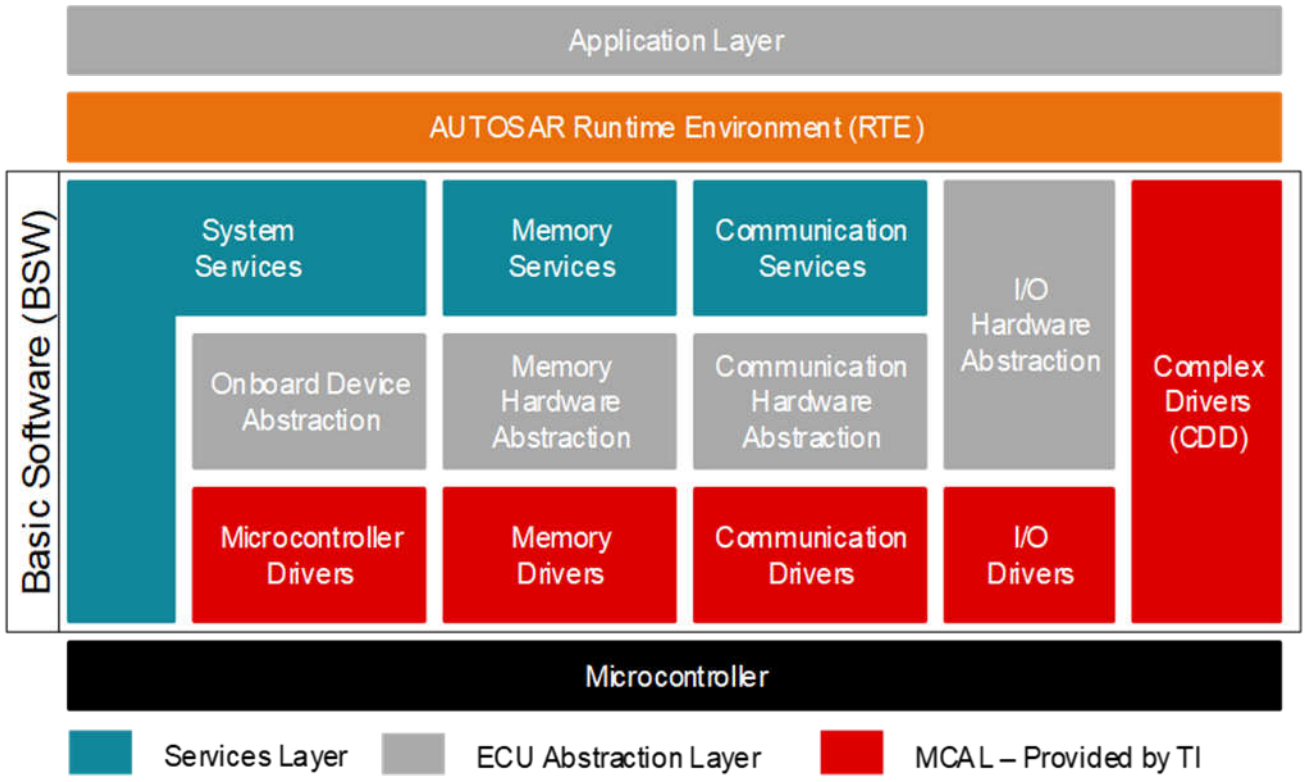

Figure 2-1. Block Diagram of AUTOSAR Stack

2.1 What are MCAL and AUTOSAR?

AUTOSAR was established in 2003 by a consortium of automotive OEMs and suppliers to create a standardized software framework for automotive systems; it defines a layered software architecture that separates application logic from hardware-specific implementation. Figure 2-1 provides a visual representation of this layered software architecture. The three principal layers of the AUTOSAR architecture are:

1. Application layer: vehicle-level functions written independently of hardware
2. Runtime environment (RTE): middleware interface enabling portable communication
3. Basic software (BSW): hardware-dependent drivers and services, including MCAL

Note

AUTOSAR offers two platform variants:

- *Classic*, targeting resource-constrained or safety-critical ECUs with static, deterministic architectures compliant with ISO 26262; a rigid yet lightweight version of the AUTOSAR standards
- *Adaptive*, designed for high-performance computing platforms supporting OTA updates, machine learning, and dynamic service-oriented architectures; typically resreved for larger processors and SoCs

The scope of this paper primarily focuses on AUTOSAR Classic, which remains the dominant platform in traditional automotive ECU development for simpler automotive microcontrollers such as MSPM0 and MSPM33.

MCAL occupies the lowest position within the BSW layer, sitting directly above the microcontroller hardware and providing *standardized, hardware-independent APIs for core peripherals*, including ADC, PWM, SPI, CAN, LIN, GPIO, timers, interrupt controllers, and memory and clock management units. By abstracting hardware-specific details behind uniform interfaces defined by AUTOSAR specifications, MCAL makes sure that upper software layers remain fully portable across different MCU families. MCAL drivers are typically supplied by microcontroller vendors and are pre-validated to reduce both development effort and safety certification burden. Texas Instruments provides MCAL support for Arm® Cortex®-based platforms *MSPM0* and *MPSM33*, enabling AUTOSAR Classic compliance on these devices.

As shown in Figure 2-1, complex device drivers (CDDs) are a key component of the MCAL layer, extending from MCU hardware up to the AUTOSAR runtime environment. These TI-proprietary modules are not defined by the MCAL standard and expose hardware features that are not described by the standard MCAL framework. However, these modules play a key role by enabling API-access to these peripherals at the application layer.

Together, AUTOSAR and MCAL form a complementary pair: AUTOSAR defines the architectural framework and interface standards, while MCAL provides the hardware-specific implementation that makes those standards realizable on a given MCU.

2.2 MCAL Applications

MCAL is primarily used in automotive applications where standardized hardware abstraction is essential, including chassis and safety systems, thermal management, interior/exterior lighting, and body electronics. This software has also been adopted in some industrial applications where similar portability and determinism requirements apply.

A few key considerations drive MCAL adoption:

1. Hardware independence: By isolating application code from microcontroller-specific details, MCAL allows the migration of vehicle systems to new MCU families with minimal software rework, reducing development risk and cost when scaling designs across platforms or product generations.
2. Functional safety support: MCAL drivers are pre-validated and documented against AUTOSAR specifications, directly supporting ISO 26262 compliance efforts and reducing the certification overhead that can otherwise burden individual project teams.
3. Real-time determinism and vendor flexibility: MCAL provides predictable, low-latency access to hardware peripherals while enabling integration with multiple third-party hardware and software vendors, avoiding dependence on a single supplier ecosystem.

Notably, MCAL's value is not contingent on deploying a full AUTOSAR stack. Developers can adopt MCAL's standardized hardware abstraction layer independently, retaining flexibility in their upper-layer architecture while still benefiting from pre-validated drivers, reduced development effort, streamlined code portability across projects. Consequently, MCAL can be a practical choice even for teams operating outside a strict AUTOSAR environment, especially for quickly iterating designs or when developing software in resource-constrained scenarios.

2.3 AUTOSAR Applications

AUTOSAR serves as the backbone of modern automotive software development. The safety-compliant architecture and deterministic scheduling of AUTOSAR are essential for mission-critical systems like powertrain and chassis control. For ADAS and autonomous driving, AUTOSAR enables structured sensor fusion and efficient high-bandwidth data management. Furthermore, body electronics and lighting applications benefit significantly from the cost advantage afforded by adopting this standardized software architecture. Factors such as functional safety, determinism, and multi-vendor interoperability are key reasons why OEMs can mandate AUTOSAR implementation for a given ECU or vehicle system.

From the OEM perspective, mandating AUTOSAR compliance across the supply chain is a risk mitigation and cost control strategy. A shared architectural standard provides a common language for interface definition and integration, reduces vendor lock-in, and simplifies the engineering effort required to incorporate new components into existing vehicle platforms. Additionally, the collaborative structure of the AUTOSAR consortium makes sure the standard evolves in step with industry challenges while remaining implementable across diverse organizations and technology stacks.

3 MCAL Package for MSP Microcontrollers

Texas Instruments provides a comprehensive AUTOSAR Classic 4.3.1 compliant MCAL package designed to accelerate automotive software development across the MSP MCU portfolio, including both Arm Cortex-M0+ and Arm Cortex-M33 platforms. The package covers the full scope of standard AUTOSAR MCAL modules, spanning *communication protocols* (CAN, LIN), *I/O control* (GPIO, PWM), *timing*, *analog processing* (ADC), and *memory and clock management*; alongside these standard MCAL modules are unique *Complex Device Drivers (CDDs)* that expose TI-specific hardware capabilities not addressed by standard AUTOSAR specifications. CDD's for MSP devices include *I2C*, *UART*, and *DMA*. CAN drivers support both standard CAN and CAN-FD protocols with full protocol stack integration, while LIN drivers support both commander and target modes, enabling flexible network configurations. All drivers are pre-validated, compatible with AUTOSAR Classic BSW stacks, and integrate with industry-standard toolchains. The MSP-MCAL package is compliant with ISO 26262 up to the ASIL-B level.

3.1 Licensing Model

Texas Instruments' MSP-MCAL package is delivered as a fully royalty-free driver library. In an industry where MCAL and BSW licensing costs from third-party vendors can represent a significant and recurring per-unit expense, TI's royalty-free model eliminates that burden entirely. For the default, as-is MCAL package offering, there are no per-unit fees at any production volume, no recurring licensing costs, no royalty obligations, and no licensing caps or thresholds of any kind. Customers can scale from prototype to full mass production without incurring any additional software licensing costs from TI, making total cost of ownership both lower and fully predictable from the outset.

This royalty-free model has a concrete impact on program economics. For high-volume automotive programs where ECU production runs can reach millions of units, the absence of per-unit MCAL licensing fees represents direct, scalable cost savings that compound with production volume. Combined with the reduction in development and certification effort that pre-validated MCAL drivers provide, the TI MCAL package delivers both upfront engineering savings and long-term production cost advantages, without compromising on AUTOSAR compliance, safety certification support, or hardware abstraction quality.

3.2 Support Model

TI provides comprehensive direct technical support for the MCAL layer through the MCAL-to-BSW integration boundary. Within this scope, TI covers MCAL driver functionality per AUTOSAR 4.3.1 specifications, configuration through EB tresos® plugins, API compliance with AUTOSAR SWS documents, bug fixes, device-

specific errata workarounds, performance optimization, and memory footprint guidance for MCAL configuration. TI also provides a level of best-effort guidance for integration of TI MCAL drivers into major third-party BSW stacks, providing customers with flexibility in their choice of AUTOSAR stack while maintaining a well-supported integration path.

Support beyond the MCAL-to-BSW integration boundary is explicitly outside TI's scope. BSW module development and support (for example, CanIf, ComM, NvM, DEM, DCM) are the responsibility of the chosen AUTOSAR BSW vendor or integrator. RTE generation and configuration, application software development, and overall system integration, validation, and ISO 26262 system certification remain the responsibility of the end-developer.

4 MSP Scalability and Ecosystem

Texas Instruments provides a unified software foundation designed to bridge the gap between cost-optimized entry-level microcontrollers and high-performance, security-focused platforms. By integrating MCAL into the MSP product ecosystem, TI enables a seamless transition between different processor architectures. This allows developers to maintain a consistent software approach as project requirements evolve from simple control tasks to complex, safety-critical automotive or industrial applications.

4.1 Platform Scalability

The MSP-MCAL package enables platform scalability through a streamlined abstraction layer that spans both MSPM0 and MSPM33 architectures. Currently, there are individual packages for both MSPM0 and MSPM33. Each MSP-MCAL software package allows developers to reuse application code across both cost-optimized and high-performance device variants, drastically minimizing engineering rework when migrating designs between different product series.

Two core pillars drive this scalability. First, *software portability* is achieved through standardized peripheral interfaces that abstract hardware-specific details, making sure that the application layer remains independent of underlying implementation differences. Second, *hardware consistency* across the MSP product series, both at the IP-level and across pin-to-pin package options, reduces the barrier to entry and simplifies migration paths. Together, these capabilities accelerate time-to-market, lower overall development costs, and allow for seamless product line scaling from entry-level designs to high-performance, feature-rich systems.

4.2 Software Ecosystem and Integration

The overarching MSP microcontroller ecosystem is primarily comprised of the MSP SDKs, which provide driver libraries, middleware, code examples, and RTOS support for general embedded development. Within this ecosystem, MCAL serves a specialized role by providing the hardware-dependent drivers that abstract peripheral access, allowing higher-layer software to remain completely hardware-independent.

It is important to distinguish that *MSP-MCAL is a separate package* that is distinct from the standard MSP SDKs. While the SDK is intended for general-purpose embedded development, the MCAL package is specifically designed for AUTOSAR-compliant applications.

5 Getting Started

Successfully integrating MCAL into your automotive software stack requires careful planning regarding toolchains, memory allocation, and architectural boundaries. This section provides some high-level guidance on locating the package, setting up your development environment, and optimizing your system resources.

5.1 Installation and Setup

The MSP-MCAL package can be downloaded as a secure resource from TI.com. For detailed guidance during installation and setup, please refer to the MCAL User Guide included in the MCAL package download.

To provide predictable behavior and optimal code generation, the MCAL is validated with industry-standard compilers. Users must make sure their toolchain supports the necessary optimization flags (such as function-section elimination and short-enum support) to allow the linker to strip unused code, which is vital for maintaining a lean memory profile in resource-constrained automotive ECUs.

The MSP-MCAL package is highly modular, allowing developers to tailor the software footprint to the specific requirements of the ECU. Rather than enabling the entire suite, developers must only activate the specific modules required by the application. Furthermore, the choice between PRE-COMPILE and POST-BUILD configuration variants must be used strategically to balance the need for runtime flexibility against the requirement for minimal flash consumption.

Note

TI recommends the use of Elektrobit's EB tresos tool for evaluating and developing with the MSP MCAL package. Installation and setup of this tool requires first acquiring a license from TI. Please refer to the MCAL User Guide for more information.

5.2 Implementation Tips

Effective memory management is critical to making sure that an automotive system can accommodate both primary functions and future updates (such as OTA updates or diagnostic expansions).

To maintain a scalable architecture and optimize the code footprint, developers must employ several key optimization techniques:

- **Selective Module Activation:** Only enable drivers that are necessary for the specific peripheral set being implemented in the design.
- **Optimization Flags:** Utilize compiler-specific flags for section elimination and data compression to minimize the impact of the code on flash and RAM.
- **Feature Tuning:** Where permitted by the AUTOSAR specification, disable non-essential diagnostic or error-detection features (such as DET) in production builds to reclaim memory.
- **DMA Utilization:** For high-bandwidth peripherals like SPI or ADC, utilize DMA-enabled driver variants to optimize CPU utilization and throughput, while accounting for the incremental increase in RAM allocation.

When performing a feasibility study for a new design, engineers must establish a "memory budget" that accounts for the MCAL, the chosen AUTOSAR BSW stack, the application logic, and a significant buffer for calibration data and bootloader requirements. A well-advised practice is to target a flash utilization threshold that leaves sufficient headroom for field updates, maintaining the longevity of the product throughout its life cycle.

Resources

- MSPM0 MCAL Package Download
- MSPM33 MCAL Package Download
- EB tresos® Tool Downloads Page
- EB tresos® License Request

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale, TI’s General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Last updated 10/2025