Enabling Differentiation through MCU Integration on Jacinto™ 7 Processors

TEXAS INSTRUMENTS

Mahmut Ciftci
Systems Architect,
Jacinto Processors

Yashwant Dutt
Engineering Manager,
Jacinto Processors

Sujith Shivalingappa
Engineering Manager,
Jacinto Processors

Automotive architectures have significantly changed over the last decade to meet consumer demand for more entertainment, connectivity and functional safety features. The recent focus on autonomous driving has turned cars into an innovation hub and has put them at the forefront of technological advances. As a result, system complexity, semiconductor content and costs have increased exponentially.

Today, a typical car has hundreds of electronic control units (ECUs) to manage various functions. Most of these ECUs are simple microcontrollers (MCUs). However, more complex systems such as advanced driver assistance systems (ADASs) and automotive gateway systems require more powerful application processors in addition to a vehicle MCU, each of which performs certain system functions.

This white paper describes the role of the vehicle MCU in automotive systems and presents the silicon architecture of Jacinto™ 7 processors, which integrate the vehicle MCU into the application processor.

Vehicle MCUs in automotive systems

Complex automotive embedded systems split computing responsibilities between the application processor and the vehicle MCU (also known as the wake-up MCU). **Figure 1** shows a block diagram of such a system, where the application processor could be a single or multicore processor hosting single or multiple high-level operating systems, as well as driving displays; handling application software and middleware; and managing high-throughput data, complex graphics, cameras and vision processing.

The MCU is responsible for all actions taken by the ECU. It monitors the application processor and evaluates its computational results. The MCU also processes other sensor inputs, manages communication with the vehicle network (through Controller Area Network [CAN], Local Interconnect Network [LIN] and Ethernet), performs other housekeeping activities, and supports wake-up and standby functionalities Depending on the functional safety use case(s), the vehicle MCU can also manage functional safety requirements.

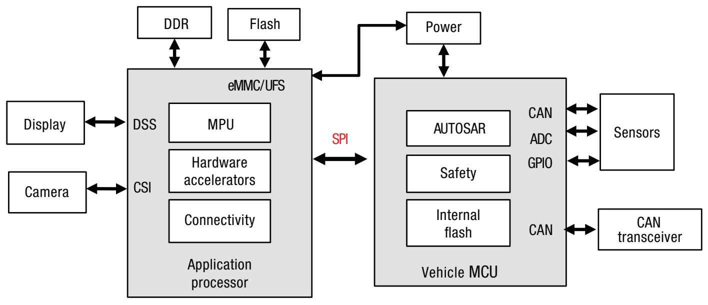

Figure 1. *A generic block diagram for a complex automotive system.*

A typical vehicle MCU supports:

- Diverse connectivity interfaces to communicate with other nodes of a vehicle network through CAN, LIN and Ethernet.
- Diverse input and output interfaces (such as general-purpose inputs/outputs [GPIOs], analog-to-digital converters [ADCs], Serial Peripheral Interface [SPI] and I2C) to monitor and control various sensors and peripherals.
- High-speed memories to store data and programs (embedded/external memories).

- Low-power standby modes.
- Fast boot up to respond to vehicle network messages (such as a CAN response within 50 to 100 ms).
- Automotive Safety Integrity Level (ASIL)-D safety support, if required, to meet system safety goals.
- The Automotive Open System Architecture (AUTOSAR) stack and applications.

As system complexity and feature sets increase, so do the requirements of the vehicle MCU, including:

- Increased computing power to satisfy the higher computing needs of complex software.
- Increased embedded flash sizes to handle more complex software.
- Increased high-speed memories to accommodate large software (both data and instructions).
- Multiple Ethernet ports to support multiple networks.
- A higher number of input and output interfaces (CAN, LIN and ADCs).
- Higher cybersecurity requirements.

These increased requirements mean higher costs for vehicle MCUs and therefore increased bill-of-materials (BOM) costs.

Software development is a challenge with the split architecture shown in **Figure 1**. The application processor and vehicle MCU are based on different architectures and have their own respective software development kits (SDKs). Software development and validation must occur in two different software environments, significantly increasing software development and validation complexity and effort.

A split architecture also complicates the ability to meet functional safety and security system requirements. Managing both the application processor and the MCU becomes challenging if you have to take care of requirements for both components. Also, in a system with an ASIL-D safety requirement, interprocessor communication (IPC) requires significant central processing unit performance to meet functional safety needs (especially for large amounts of data).

Jacinto DRA8xx and TDA4xx systems-on-chip (SoCs) are part of the Jacinto 7 processor family and provide a novel architecture that integrates the vehicle MCU into an application processor. This architecture addresses ever-increasing system requirements while optimizing system BOM costs, unifying software development, and simplifying functional safety and security support.

MCU integration on Jacinto 7 processors

The Jacinto 7 automotive application processor platform includes innovative features for ADASs, automotive gateways and cockpit systems – especially with the integration of the vehicle MCU. Figure 2 shows the high-level architecture of a Jacinto 7 SoC. The application processor is divided into two independent domains: the main domain and the MCU domain. The main domain provides high-performance computing cores such as a microprocessing unit and graphics processing unit, multimedia accelerators, and vision hardware accelerators including digital signal processors. The main domain also provides the necessary input and output and video interfaces (such as capture and display).

The MCU domain replaces the functionality typically offloaded to an external vehicle MCU. The main domain and MCU domain are separated from each other with independent voltage, power, clock and reset functions. Hardware firewalls guarantee freedom from interference (FFI) between the two domains.

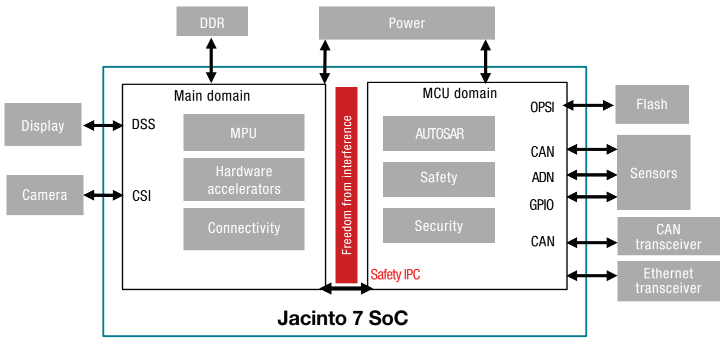

***Figure 2**. Jacinto 7 SoC-based system architecture.*

The Jacinto 7 SoC's MCU domain includes:

- **Processing cores.** The MCU domain is based on dual Arm® Cortex®-R5F processors; each core is configurable to either lock-step or split modes. Cortex-R5F cores run up to 1 GHz and provide significantly higher performance compared to stand-alone vehicle MCUs.
- **Peripherals and I/O interfaces.** The MCU subsystem has an extensive set of I/Os to support automotive use cases, including:
  - Multiple instances of CAN-Flexible Data Rate.
  - Ethernet.
  - Multiple instances of GPIO, SPI, I2C and pulse-width modulation.
  - Multichannel ADCs.
- **Functional safety.** The Jacinto 7 SoC MCU domain is designed to support systems up to ASIL-D. The following functional units have been integrated into the MCU domain enables a higher safety rating:
  - Built-in self-tests.
  - Error correction code (ECC) on all memories.
  - Error signaling modules.
  - Cyclic redundancy checks.
  - Watchdog timers.
  - Dual clock comparators and temperature sensors.

For more details, see our Leverage Jacinto™ 7 processors functional safety features for automotive designs white paper.

- **Security.** The MCU domain of the Jacinto 7 architecture is the security master of the whole SoC. The MCU domain includes a device management and security controller that provides:
  - Secure boot up with a unique key.
  - Cryptoaccelerators: Rivest, Shamir and Adelman-4K; true random number generator/deterministic random bit generator; Secure Hash Algorithm 2-512; and Advanced Encryption Standard-256.
  - Hardware security module services.
  - Memory and peripheral firewalls.

Power

Low standby power is one of the key features offered by the vehicle MCU. The Jacinto 7 SoC achieves low standby power by providing independent power to the MCU domain. In a typical scenario, the MCU domain will be powered off and wake up only when there is CAN activity. Depending on the CAN message received, the SoC will either initiate full system power-up or go back to off mode.

Flash

The Jacinto 7 SoC does not support integrated flash and instead relies on external NOR flash, such as Octal SPI (OSPI) or Hyperflash, to store boot and other images. The MCU domain includes internal random access memory with ECC support to run the AUTOSAR stack or other software and has access to large external double-data-rate memory for additional program and data space. Execute in place (XIP) is supported on OSPI to enable shorter wake-up times, and the image in XIP is authenticated before it starts running.

Boot up and early CAN response

The fast boot time for early CAN response is achievable on the Jacinto 7 processor platform with MCU integration. The MCU domain is the boot master of the full SoC and can boot up and run the CAN stack to meet 50ms to 100ms requirement.

Comparing Jacinto 7 MCU integration with an external vehicle MCU

Table 1 compares features of the Jacinto 7 platform's MCU domain to an external MCU.

Features	Jacinto 7 MCU	External MCU
Processing cores	Provides higher performance; cores can run at higher speeds compared to an external MCU	Lower performance
I/O support	CAN, ADC, SPI, GPIO, PWM, Ethernet	CAN, ADC, SPI, GPIO, PWM, Ethernet
Functional safety	Up to ASIL-D; simplifies safety support for mixed-criticality applications	Up to ASIL-D; can lag behind latest functional safety requirements
Security	Latest security support	Can lag behind latest security requirements
Power	Can meet low-power requirements	Low standby current
System BOM cost	Provides significant system BOM cost savings through the removal of an external MCU, and save printed circuit board (PCB) area	Can result in significant cost increases depending on flash size,functional safety requirements, etc.
Boot	Meets 50- to 100-ms boot times for CAN response	Meets 50- to 100-ms boot times for CAN response
Software development	Unified software development with application processor	Separate SDKs
Flash	External flash (OSPI, Hyperflash)	Internal flash; cost of MCU increases significantly as more flash size is needed.
Communication	Internal IPC provides faster and safer communication	External interface, SPI, etc.

Table 1: *Features of Jacinto 7 processors with MCU integration compared to traditional external MCUs.*

The Jacinto 7 software architecture

Jacinto 7 software development, including development for DRA8xx and TDA4xx SoCs, is unified for both the application processor and the vehicle MCU. A single Jacinto 7 platform SDK enables software development for both the main domain and the MCU domain.

Figures 3 and 4 show example software architectures for gateway and ADAS use cases. In these examples, the MCU domain runs the:

- System boot up and device management.
- AUTOSAR real-time operating system, stack and applications.
- Diagnostics.
- Functional safety and security services.

The main domain runs the:

- High-level operating systems.
- Applications such as adaptive AUTOSAR and customer applications.
- Middleware and connectivity.
- Vision and multimedia algorithms.

A high-performance IPC that is functional safety capable manages communication between the main domain and the MCU domain.

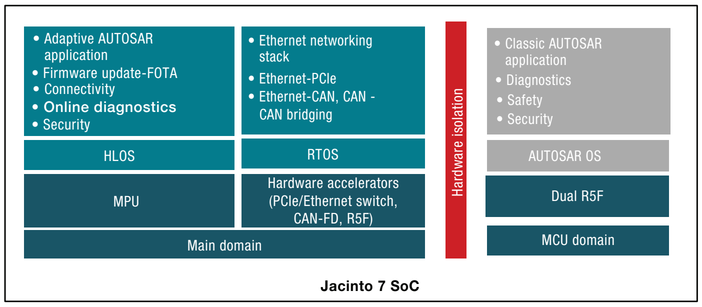

Figure 3. *Jacinto 7 software architecture for a typical gateway system.*

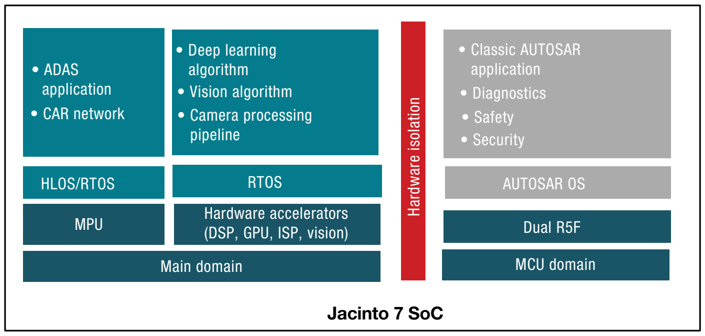

Figure 4. *Jacinto 7 software architecture for an ADAS.*

Conclusion

Integrating the vehicle MCU into the Jacinto 7 platform provides many advantages compared to an external MCU, including:

- A flexible and high-performance MCU.
- Lower system BOM costs.
- Unified software development.
- Simplified functional safety and security support.

In addition, vehicle MCU integration is common across all Jacinto 7 family products, which scale across various end-equipment types to enable software and hardware reuse.

For more information on Jacinto 7 processors, please visit ti.com/jacinto7.