Inside Tomorrow's SDVs: Integrating Remote-Controlled Edge Nodes

TEXAS INSTRUMENTS

Kate Hawkins
Systems Engineer
Body Electronics and Lighting
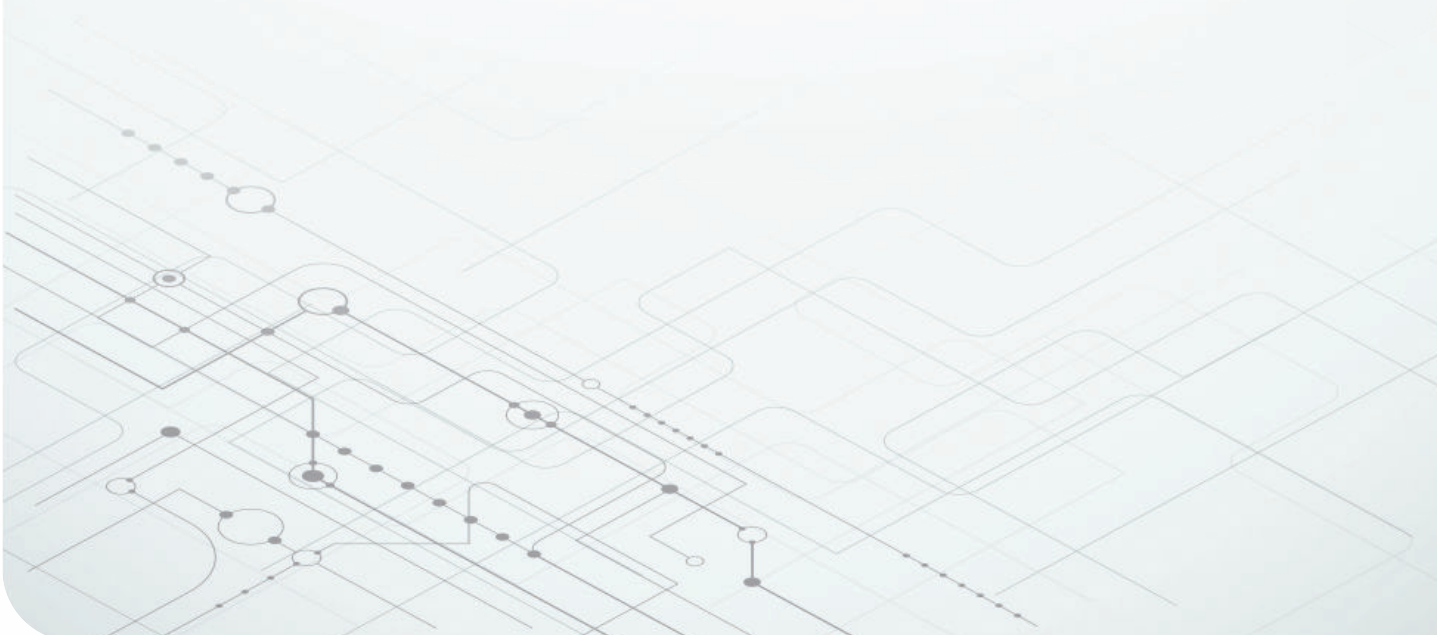
Remote-controlled edge technology transforms automotive networks, enabling a more centralized architecture for SDVs.

At a glance

📄 **1**   **Traditional vs. remote-controlled edge nodes**

📄 **2**   **Remote-controlled edge-node benefits**

📄 **3**   **Remote-controlled edge-node considerations**

Figure 1. An automotive zone architecture with commander ECUs and multiple edge nodes.

Introduction

Automotive in-vehicle networking is evolving to support new features in software-defined vehicles (SDVs). As software consolidates into fewer electronic control units (ECUs) to increase scalability across vehicle platforms and streamline over-the-air (OTA) updates, a new remote-controlled edge concept optimizes wiring while enabling scalable edge-node software.

Edge nodes are specialized ECUs that handle the real-time control of specific functions, such as headlight modules for exterior lighting or control modules for door locks, windows and side mirrors. These nodes receive commands from a commander ECU (the zone controller, domain controller or central computing) throughout the in-vehicle network. Edge nodes manage local hardware control by monitoring temperature, pressure or position sensors for control-loop feedback while directly controlling mechanical actuators such as motors and solenoids through load drivers, including half bridges and high- and low-side switches. **Figure 1** illustrates the difference between an edge node and commander ECUs in a zone architecture.
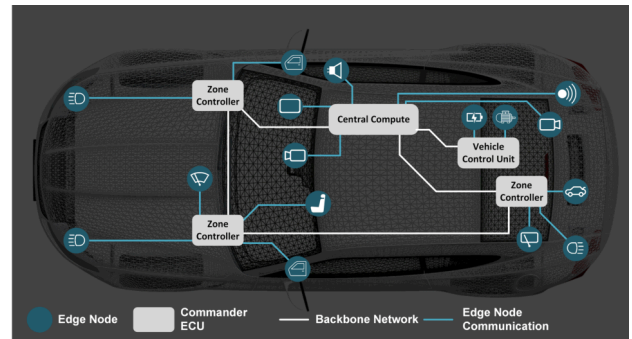
Remote-controlled edge architectures shift the real-time control and hardware abstraction layer (HAL) upstream to the commander ECU, which then generates low-level hardware commands for sensors and load drivers to the edge nodes. The remote-controlled edge solution bridges the higher-level network data-link layers between ECUs such as Ethernet or Controller Area Network (CAN) with low-level communication interfaces such as Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I2C), universal asynchronous receiver-transmitter (UART) and general-purpose input/output (GPIO). This approach eliminates the microcontroller (MCU) as well as any software from the edge node entirely.

The remote-controlled edge scheme supports major, overarching trends around SDVs and reduces the amount of wire harnesses by centralizing software in the commander ECU, while keeping load-dependent hardware in the edge nodes close to the electromechanical actuators.

To learn more about SDVs, read the white paper, **Software-Defined Vehicles Shift the Future of Automotive Electronics Into Gear**.

Traditional vs. remote-controlled edge nodes

Figure 2 shows a traditional edge node block diagram. In the traditional architecture, the local MCU includes a HAL, which is the software that defines how device software drivers interact with the hardware. The edge MCU receives commands from the controller MCU over a network interface, typically a CAN Flexible Data-Rate (CAN FD) Local Interconnect Network, and controls local hardware based on instructions from the controller.

For example, if the upstream controller MCU sends a command to the edge MCU node to "roll driver window up," the edge MCU translates this message into specific hardware actions, including rolling the window up, performing a window soft close, and protecting against a potential motor stall or window pinch event. The edge-node MCU communicates necessary SPI messages to the motor driver and implements the real-time control loop of the window motor through pulse-width modulation (PWM) outputs to the half-bridge motor driver, while using integrated analog-to-digital converters (ADCs) to monitor the motor current and counting Hall-effect pulses for window position tracking.
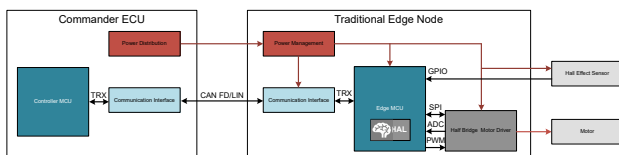

Figure 2. Block diagram of a traditional edge node with communication to a commander ECU.

Figure 3 shows a remote-controlled edge-node block diagram. This architecture moves the HAL and real-time actuator upstream into the commander ECU's MCU, eliminating the edge-node MCU entirely. The controller MCU now can send a command that includes device communication protocol frames or peripheral control (SPI, I2C, UART, PWM output control, ADC sampling or GPIO).

For window lift applications, the controller transmits the direct control data (SPI motor driver command and PWM output setting) over the network, embedded within standard communication protocol data payloads (CAN FD light or 10BASE-T1S). A communication bridge in the edge node extracts these protocol data payloads to output the SPI frames and PWM signal on the appropriate GPIO pins. For sensor feedback, this bridge samples an internal or external ADC and Hall-effect sensor data and sends it back to the commanding ECU to complete the control loop.
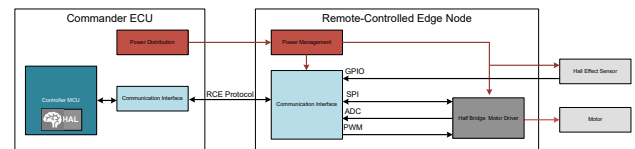

Figure 3. Block diagram of a remote-controlled edge node with communication to a commander ECU.

Remote-controlled edge-node benefits

Remote-controlled edge architectures offer multiple advantages, including centralizing software, reducing software development costs, enabling scalability, and simplifying OTA updates. Additionally, using a remote-controlled edge node enables load driver control from the commander ECU while minimizing load wiring.

A remote-controlled edge node can reduce system costs through software centralization. Removing the edge microcontroller and centralizing software into fewer ECUs enables companies to decrease the amount of software development and management overhead, decreasing testing and validation requirements across the many ECUs in the vehicle.

Software centralization also enhances scalability. Developers can create software for the upstream commander ECU only, while standardizing hardware in the edge nodes. This standardization simplifies vehicular infrastructures across multiple nodes and ECUs, rather than requiring specialized edge hardware.

Figure 4 contrasts a traditional approach (where each edge-node module uses a different MCU from a different supplier, requiring software development and management across multiple platforms) to a remote-controlled edge approach (where the label "RCE Solution A, B or C" in Figure 4 represents software-free options from multiple suppliers). Standards-based solutions provide additional benefits, as the commander ECU's software remains consistent regardless of remote-controlled edge solution supplier.
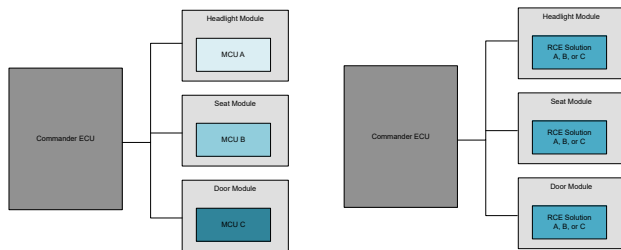

Figure 4. *Hardware scalability of a remote-controlled edge node vs. a traditional edge node.*

Centralizing control enables automakers to streamline software management and OTA updates, making it easier for them to own and manage their own software. Releasing an OTA update requires updating only the commander ECU, rather than updating the software for multiple modules.

Using edge nodes instead of driving loads directly from the commander ECU shortens wire length to the load drivers. Remote-controlled edge nodes maintain this benefit while also keeping the HAL in the commander ECU. Figure 5 shows this configuration in a zone architecture using a door as an example. Although the zone controller controls both door modules, the door edge module shortens the load cabling, which also helps mitigate electromagnetic interference by minimizing parasitic capacitance and inductance, which is especially important for next-generation 48V vehicles that require faster switching times.
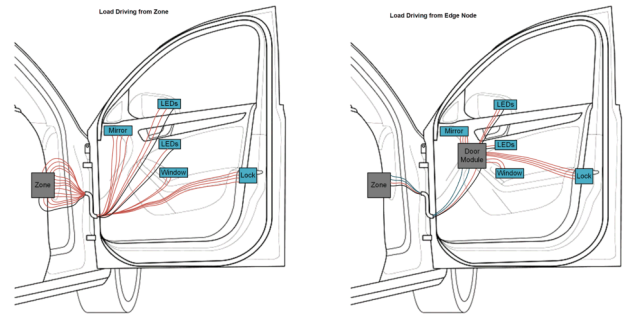

Figure 5. *Cable reduction of a remote-controlled edge node vs. a traditional edge node.*

Remote-controlled edge-node considerations

Original equipment manufacturers (OEMs) and designers investigating remote-controlled edge technology must consider latency, functional safety, cybersecurity and cost.

Latency is a significant design challenge. Data from the edge must travel upstream, where decisions are made from the edge for processing, then back downstream to the edge for implementation, which adds latency delays to the real-time control loops. Figure 6 shows this process for sensing and controlling a load. Traditional edge nodes only require Step No. 2 and Step No. 5, while remote-controlled edge solutions implement features such as intelligent actions or autonomous polling to reduce delays. Intelligent actions allow the bridge device to automatically transmit sensor data without initial prompting by the commander ECU, eliminating Step No. 1. Autonomous polling enables the bridge device to automatically sample from a sensor and store the reading to a buffer. This allows Step No. 2 to occur during other steps, which can help further reduce latency.
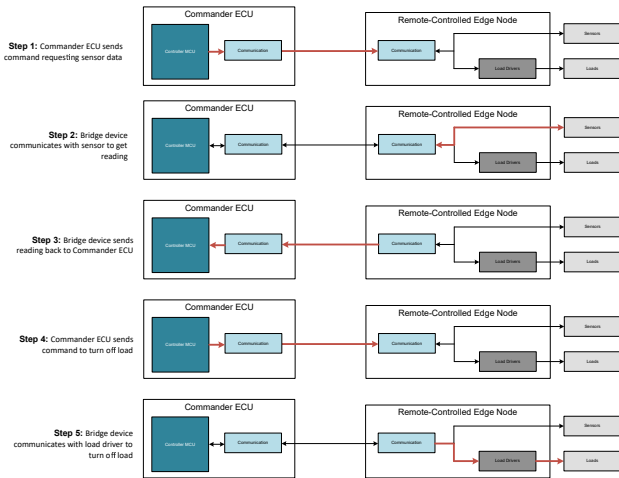
Figure 6. Communication steps of a remote-controlled edge node that contribute to latency.

Functional safety concerns may arise because there is no longer local real-time control. Edge applications with strict requirements such as tight latency requirements from Fault Tolerant Time Interval specifications may struggle with upstream communications delays. As a newer technology, first-generation remote-controlled edge devices may not meet Automotive Safety Integrity Level requirements, or may need additional measures to achieve functional safety at a system level.

Cybersecurity risks increase as vehicles become more software-dependent. Without proper security measures, hackers can access the vehicle network and control features throughout the vehicle, which can result in theft and safety risks. Cybersecurity is more difficult to implement on remote-controlled edge nodes since there is no MCU to manage security locally, so it is important for OEMs to select solutions that meet their cybersecurity needs.

Cost considerations must balance hardware and software expenses. Replacing the low-level MCUs currently used in traditional edge nodes with remote-controlled edge-node devices is potentially more expensive. It is important to remember, however, that even if hardware costs increase, there are still significant savings in software development and management costs.

Remote-controlled edge enables automakers to manage more software internally, requiring OEMs to evaluate the trade-offs.

Remote-controlled edge applications

Remote-controlled edge technology offers value across many applications, including lighting, battery management systems (BMS), advanced driver assistance systems (ADAS), car access and body motors. Table 1 lists these applications, and the benefits of remote-controlled edge nodes.

Application	Why Remote-Controlled Edge Nodes?
Headlights	Requires only a single low-level protocol (UART, SPI, or both)
Ambient lighting	
BMS	
Radar	Numerous nodes throughout the vehicle provide hardware scalability opportunities
Ultrasonic sensing	
Car access	
Seat modules	Load drivers now integrate more diagnostic features
Door modules	

Table 1. Various remote-controlled edge-node applications and why they are a good fit.

Remote-controlled edge protocols

The solutions for remote-controlled protocols include 10BASE-T1S, CAN FD light and UART over CAN. These protocols operate in half duplex, allowing non-simultaneous, bidirectional data transmission between two devices. Half duplex enables multidrop capability, where more than two devices communicate on the same bus, requiring only a single networking device in the commander ECU to interact with multiple edge nodes. Figure 7 illustrates an example of a multidrop topology.
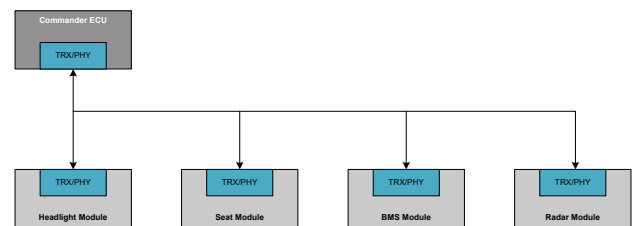

Figure 7. A multidrop topology from a commander ECU to edge nodes.

10BASE-T1S, CAN FD light and UART over CAN differ in speed, payload capacity and number of nodes in the multidrop and bus topologies. Table 2 compares these protocols.

	10BASE-T1S	CAN FD light	UART over CAN
Network protocol	Ethernet	CAN	UART
Speed	10Mbps	1-5Mbps	0.1-1Mbps
Payload	46-1,500 bytes	1-64 bytes	1-64 bytes
Maximum number of nodes	16	64	64
Topology	Round robin	Commander responder	Commander responder

Table 2. Remote-controlled edge networking protocol comparison between 10BASE-T1S, CAN FD light and UART over CAN.

Figure 8 shows the difference between the round-robin and commander-responder topologies. The round-robin topology operates cyclically, where each node has a dedicated transmission opportunity per cycle based on its node ID. This automates arbitration, but requires mediation to ensure that priority or time-critical data is not delayed by low-priority data on the bus. The commander-responder topology requires the commander ECU to prompt downstream nodes before sending data on the bus. The order of transmission is up to the commander ECU rather than being dictated by node ID.
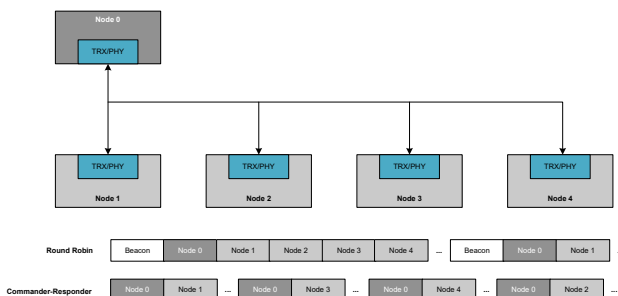
Figure 8. Transmission comparison of round-robin vs. commander-responder topologies.

10BASE-T1S, standardized by Institute of Electrical and Electronics Engineers (IEEE) 802.3cg, uses the Remote Control Protocol, which is standardized by Technical Committee 18. It operates at 10Mbps and in a round-robin multidrop topology. As an Ethernet protocol, 10BASE-T1S can incorporate Ethernet features such as Media Access Control Security (MACSec), Time-Sensitive Networking (TSN), Audio Video Bridging (AVB) and Power over Data Line (PoDL). Table 3 describes these four features. Additionally, systems already using a high-speed Ethernet backbone may benefit from simplified software with an all-Ethernet network.

Feature	Description	Standard
MACSec	Layer 2, point-to-point cybersecurity protocol for Ethernet	IEEE 802.1AE
TSN	Standards enabling deterministic, real-time communication for data synchronization throughout an Ethernet network	IEEE 802.1Q IEEE 802.1AS
AVB	Standards defining TSN for audio and video applications	IEEE 802.1BA IEEE 1722
PoDL	Power transmission over shielded twisted-pair cables used for point-to-point Ethernet	IEEE 802.1cg

Table 3. List and description of 10BASE-T1S Ethernet features and standards.

CAN FD light, a variant of CAN FD based on the International Organization for Standardization (ISO) 11898-1:2024 standard, operates at 1Mbps to 5Mbps. Unlike traditional CAN, which follows CAN arbitration (where nodes transmit simultaneously and the node with the lowest node ID wins), CAN FD light operates using a commander-responder topology. Edge nodes employ CAN FD light responders, while commander ECUs use CAN FD light commanders or CAN FD transceivers. Since many preexisting architectures already use CAN FD transceivers to communicate with edge nodes, integrating CAN FD light into current architectures is easy. Achieving speeds >1Mbps requires CAN FD light commanders, however, given controller arbitration phase constraints.

Both the 10BASE-T1S and CAN FD light protocols bridge Ethernet and CAN to other protocols such as SPI, I2C, UART, GPIO and PWM (see **Figure 9**). This bridging enables remote control of multiple sensors and drivers through 10BASE-T1S and CAN FD light, making both solutions versatile across various end applications.
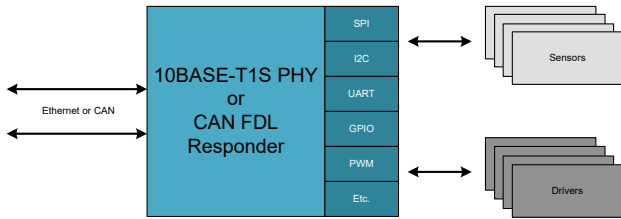
Figure 9. *Block diagram of a 10BASE-T1S or CAN FD light edge node.*

UART over CAN transmits UART packets over the CAN physical layer (PHY) using CAN transceivers (see **Figure 10**). Operating at ≤1Mbps in a commander-responder topology, UART over CAN offers a cost-effective solution but relies on UART-based drivers such as an LED, or motor drivers with integrated real-time control and diagnostic features.

Figure 10. *Block diagram of a UART over CAN edge node.*

Smart drivers with integrated real-time control complement remote-controlled edge solutions by reducing the amount of upstream control requirements. Texas Instruments (TI) offers smart motor drivers with integrated control for sensorless motor systems, including sensorless field-oriented control for brushless-DC (BLDC) motor drivers and integrated current sensing and stall detection for stepper motor drivers. Stepper motors are especially good for remote-controlled edge applications because they require less upstream diagnostic data given the increased rotation accuracy. **Table 4** lists some TI devices.

Device	Type	Field-effect transistor
MCF8329A-Q1	BLDC motor driver	External
MCF8316C-Q1	BLDC motor driver	Internal
MCF8315C-Q1	BLDC motor driver	Internal
DRV8889-Q1	Stepper motor driver	Internal

Table 4. *TI's motor-driver offerings.*

Remote-controlled edge system solutions

Figure 11 illustrates a remote-controlled edge node as a headlight using 10BASE-T1S or CAN FD light. The PHY or responder translates Ethernet or CAN FD light messages to various local protocols, controlling a temperature sensor, LED driver, motor driver and high-side switch. The commander ECU gives the PHY or responder commands to enable the load drivers through UART, SPI, GPIO or other protocols to turn the actuators on and off. The PHY or responder then transmits sensor data and actuator feedback upstream to the commander ECU.
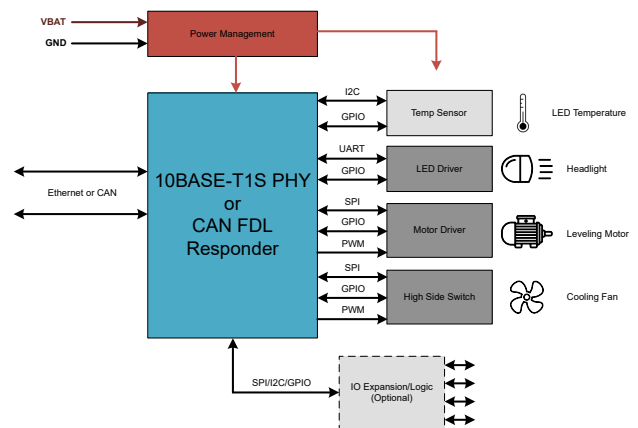
Figure 11. *Block diagram of a remote-controlled headlight module using 10BASE-T1S or CAN FD light.*

TI offers a remote-controlled edge headlight solution with UART over CAN using the **TPS92544-Q1** switching LED driver with integrated stepper-motor trapezoidal control and the **DRV8434A-Q1** stepper motor driver. The TPS92544-Q1 controls both the LED and motor through a single UART interface, making it an efficient solution for a headlight module. As shown in **Figure 12**, the CAN transceiver serves as a hardware medium for UART packets from the commander ECU.

These UART packets control the TPS92544-Q1 to enable the headlight, and drive the DRV8434A-Q1 device's stepper motion control for the leveling motor.
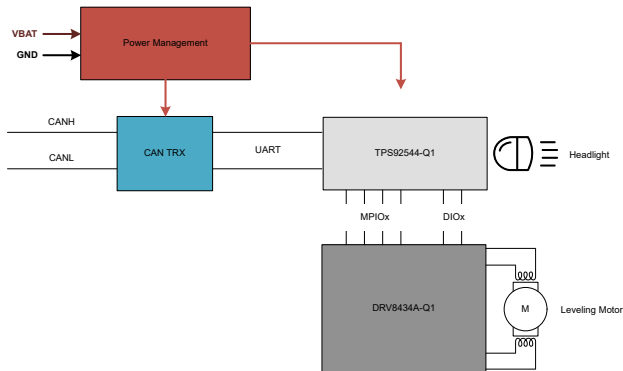

Figure 12. *Block diagram of a remote-controlled headlight module using the TPS92544-Q1 for UART over CAN.*

Conclusion

As automotive markets embrace SDVs and ECU consolidation through zone architectures, the push to centralize software will increase to allow for scalability and wire reduction. Remote-controlled edge nodes support this initiative by moving software upstream, consolidating it into fewer ECUs and simplifying OTA updates.

Multiple solutions such as 10BASE-T1S, CAN FD light and UART over CAN give system architects options for their specific design needs. Additionally, smart drivers with integrated diagnostic and control features further optimize remote-controlled edge implementations.

IMPORTANT NOTICE AND DISCLAIMER