참고 자료

차량용 전자장치의 미래를 바꾸는 소프트웨어 정의 차량

Texas Instruments

Donovan Porter
Systems Manager
Body Electronics and Lighting

Yannik Muendler
Systems Engineer
Advanced Driver Assistance Systems

이 백서에서는 영역 아키텍처를 지원하는 소프트웨어 정의 차량을 통해 더 스마트하고 안전하며 에너지 효율적인 차량을 어떻게 개발하고 있는지 알아봅니다. 소프트웨어를 중앙 집중화하고 하드웨어에서 소프트웨어를 분리함으로써 더 쉽게 업데이트하고 비용을 절감하며 새로운 기능을 사용할 수 있습니다.

한눈에 보기

머리말

자동차 OEM(Original Equipment Manufacturer)은 탑승자 경험을 개선하고, 무선 업데이트를 간소화하고, 설계 및 제조 비용을 절감하고, 더 많은 차량 데이터를 수집하고, 새로운 수익원을 창출하기 위해 지속적으로 노력하고 있습니다. 그러나 오늘날의 도메인 기반 차량 아키텍처는 이러한 요구 사항을 쉽고 효과적으로 충족할 수 없는 상태로 제작되어 **소프트웨어 정의 차량** 및 영역 아키텍처로의 전환으로 이어집니다. 소프트웨어 정의 차량은 소프트웨어를 중앙 집중화하고 소프트웨어에서 하드웨어를 분리함으로써 더 스마트하고 안전하며 에너지 효율이 높은 차량을 개발하는 다음 단계입니다.

도메인 기반 및 소프트웨어 정의 차량

오늘날의 도메인 기반 아키텍처는 자동차 제조업체가 무선 업데이트를 통해 쉽게 유지할 수 있는 확장 가능한 소프트웨어를 제공하는 데 비효율적입니다. 도메인 아키텍처는 **그림 1**에 나와 있는 것처럼 차량 내 인포테인먼트 및 ADAS(첨단 운전자 보조 시스템) 같은 도메인으로 차량 기능을 제어합니다.

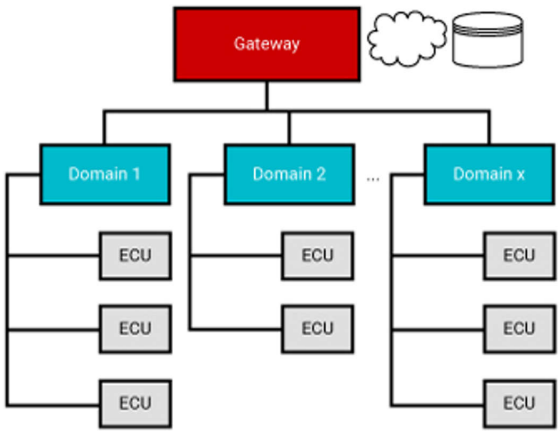

그림 1. *차량의 도메인 기반 아키텍처 다이어그램.*

차량 기능의 제어를 분할하면 여러 도메인 간의 통신과 제어가 필요한 기능을 위한 소프트웨어 개발이 복잡해집니다. 이러한 시스템의 소프트웨어를 업데이트하는 것은 모두 서로 다른 반도체 공급업체의 다양한 프로세서와 마이크로컨트롤러를 사용하는 서로 다른 계층 1 공급업체에 의해 설계 및 제조되기 때문에 어렵습니다. 차량 기능을 제어하기 위한 소프트웨어도 하드웨어와 긴밀하게 연결되어 있습니다. OEM은 각 ECU 마이크로컨트롤러에서 실행되

는 애플리케이션별 펌웨어를 통해 특정 기능(시트 조정, 주차 지원)을 수행하기 위해 ECU(전자 제어 장치)를 설치합니다. 이러한 ECU는 차량 모델과 트림에 따라 다르므로 제조 및 설계 비용이 높아집니다. 따라서 모든 차량 모델, 트림 및 개별 ECU에서 소프트웨어를 관리하는 것은 상당한 노력이 필요하며, OEM은 여러 계층 1, 심지어 반도체 공급업체들과 협력하여 새로운 소프트웨어 업데이트를 구현해야 합니다.

반면, 영역 아키텍처를 채택하는 소프트웨어 정의 차량은 소프트웨어를 중앙 집중화하여 OTA 업데이트를 단순화하고, 차량 하드웨어를 더 높은 계층 애플리케이션 소프트웨어에서 분리하고 차량 모델 및 트림 전반에 걸쳐 더 비용 효율적인 확장성을 제공함으로써 소프트웨어를 통해 새로운 기능을 추가할 수 있는 유연성을 지원합니다.

그림 2에서는 중앙 컴퓨팅 시스템에 소프트웨어를 중앙 집중화하고, 영역 제어 모듈을 구현하여 데이터를 집계하고, 부하를 작동시키고, 전력을 로컬로 분배하는 영역 아키텍처의 예를 보여줍니다. 영역 아키텍처에 대한 자세한 내용은 " **영역 아키텍처가 완전한 소프트웨어 정의 차량을 위한 기반을 다지는 방법** "을 참조하십시오.

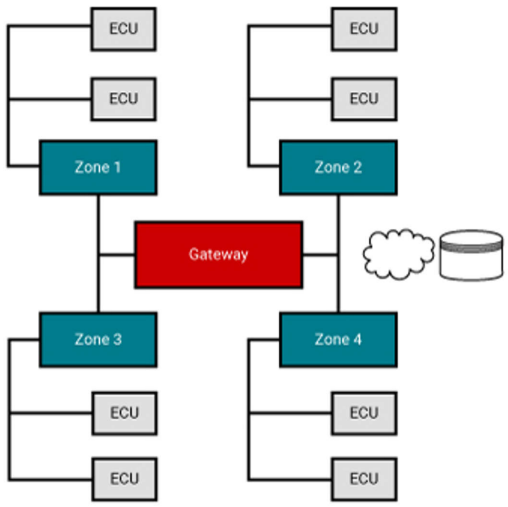

그림 2. *차량의 영역 아키텍처 다이어그램.*

소프트웨어 정의 차량에서 중앙 집중식 소프트웨어의 가장 큰 장점은 애플리케이션 소프트웨어를 호스팅하는 ECU를 줄여서 펌웨어 변경이 필요한 프로세서 및 마이크로컨트롤러의 수를 줄임으로써 OTA 업데이트를 간소화하는 것입니다. 새로운 기능과 애플리케이션을 추가하려면 중앙 컴퓨터 또는 영역 제어 모듈 소프트웨어만 업데이트해야 합니다. 다운스트림 센서와 기계적 작동을 제어하는 나머지 ECU(헤드라이트, 도어 모듈, 오디오 증폭기)는 애플리케이션 소프트웨어에서 추상화되기 때문입니다. 따라서 차량 네트워크의 엣지에서 기계적 작동을 수행하는 ECU와 센서는 덜 복잡한 펌웨어가 필요하며 향후 실시간 제어를 중앙 컴퓨터로 완전히 전환할 수 있습니다.

또한 원래 특정 애플리케이션을 위해 설계된 센서와 액추에이터의 용도를 변경하여 새로운 기능을 만들 수도 있습니다. 예를 들어 처음에 재실 모니터링용으로 설계된 침입자 또는 도난 감지와 안전 벨트 리마인더 기능을 제공하기 위해 실내 레이더 센서에 대한 새로운 애플리케이션을 추가하는 방법이 있습니다. 기본적으로 OEM은 이미 차량 내에 존재하는 하드웨어 및 센서를 통해 새로운 기능을 구현할 수 있는 더 높은 유연성을 가지고 있습니다.

마지막으로, **그림 3**에서 보듯이, 소프트웨어는 모든 차량 플랫폼에 걸쳐 확장하여 개발 비용을 더욱 절감할 수 있습니다. 이코노미 레벨 차량은 원격 키리스 엔트리, 창문 리프트 및 후방 시야 카메라와 같은 기능을 위한 고급 브랜드와 동일한 소프트웨어를 구현할 수 있습니다.

고급 모델은 기본 기능 위에 소프트웨어를 통해 프리미엄 기능을 제공할 수 있습니다. 하드웨어 변경이 여전히 필요할 수 있지만, 전반적인 접근 방식은 모듈식이며 차량 전체에 걸쳐 확장 가능합니다. 프로세서 및 마이크로컨트롤러를 추가하거나 제거하면 중앙 컴퓨터 또는 영역 제어 모듈에서 컴퓨팅 성능을 높이거나 낮출 수 있습니다.

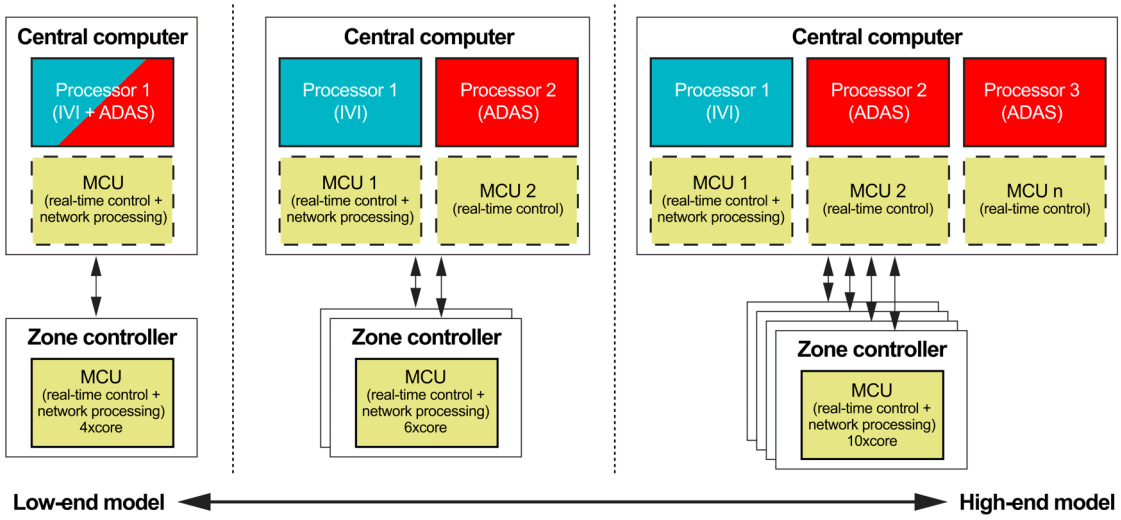

그림 3. *로우엔드 및 하이엔드 차량 모델 간의 컴퓨팅 성능 비교.*

차량 내에 시트 마사지, 스티어링 휠 난방, 도로 잡음 제거 등의 기능을 구현하려면 추가적인 하드웨어가 필요합니다. 하지만 이 추가 기능을 제어하기 위해서는 중앙 컴퓨터나 영역 제어 모듈에 대한 소프트웨어만 업데이트하면 됩니다. 새로운 MCU 없이 작동하는 기술을 이용하면 설계자가 감지 및/또는 기계적 작동을 관리하는 ECU에서 소프트웨어를 간소화하거나 제거하는데 도움이 될 수 있습니다. 예를 들어 SPI(직렬 주변 기기 인터페이스)를 사용하는 온도 센서는 SPI가 지원되는 MCU 없는 통신 PHY를 사용해 직접 통신할 수 있게 됩니다. 이 경우 MCU 없는 PHY가 MCU를 대체하며, 통합형 CAN이나 이더넷 트랜시버가 있어 MCU가 SPI를 CAN 신호로 변환할 필요가 없고, 이 센서와 통신하는 데 일반적으로 필요한 소프트웨어도 필요하지 않게 됩니다.

하드웨어 추상화 계층을 사용해 사용자 정의 차량 지원

차량에서 하드웨어 디커플링을 지원하려면 소프트웨어와는 다른 추상화 계층이 필요합니다. 표준화된 API(애플리케이션 프로그래밍 인터페이스)는 다양한 추상화 계층 간 통신을 지원하여 애플리케이션 소스 코드를 여러 개의 분산된 ECU에서 다시 사용할 수 있도록 합니다. 가장 낮은 추상화 수준은 MCAL(마이크로컨트롤러 추상화 계층)입니다.

MCAL은 SDV에서 중요한 역할을 하며, 기본 하드웨어 주변 기기의 복잡성을 추상화하는 API를 제공합니다. 이 계층은 **TDA4VH-Q1** 프로세서와 같은 중앙 컴퓨팅 SoC에 통합된 하드웨어(예: 타이머, ADC, 이더넷 서브시스템 등)와 상위 소프트웨어 계층을 연결하는 브리지 역할을 합니다. MCAL를 사용하면 애플리케이션 소프트웨어가 특정 하드웨어 세부 정보와 연계되지 않고도 하드웨어와 상호 작용할 수 있습니다. 이 추상화는 OEM에서 소프트웨어 구성요소를 최소한의 수정만으로 여러 모델과 버전에서 다시 사용할 수 있도록 하므로 다양한 차량 플랫폼에서 소프트웨어 이식성을 실현하는 데 매우 중요합니다.

상위 소프트웨어 및 MCAL 간의 인터페이스 역할로는 ECUAL(ECU 추상화 계층)이 있습니다. ECUAL은 표준화된 API를 통해 MCU와 주변 기기 장치(예: CAN 트랜시버, 이더넷 PHY 및 SerDes 장치 등) 등 이용 가능한 모든 ECU 하드웨어에 상위 소프트웨어에 대한 액세스를 제공합니다.

소프트웨어 정의 차량을 통해 새로운 기술 구현

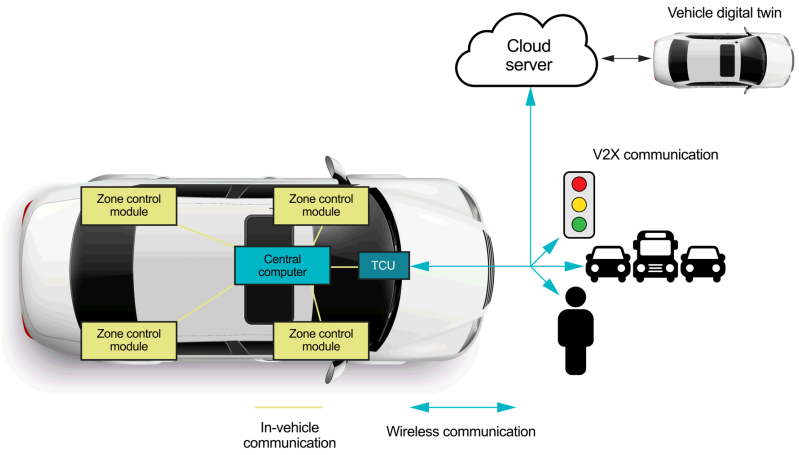

그림 4. *클라우드 및 V2X에 대한 소프트웨어 정의 차량 연결.*

소프트웨어 정의 차량은 OEM을 위한 새로운 기술과 수익원을 가능하게 합니다. 차량이 계속해서 더 많은 전자 장치와 센서를 통합함에 따라 차량 성능, 고장 시나리오 및 운전자 선호 데이터가 그 어느 때보다 많아지고 있습니다. 소프트웨어 정의 차량은 수집 과정을 간소화하고 차량 데이터를 안전하게 공유함으로써 디지털 트윈 및 V2X(차량 및 사물 간 통신) 기능을 더욱 개선합니다(**그림 4** 참조).

디지털 트윈 기능(실제 시스템의 가상 표현)을 통해 소프트웨어 정의 차량은 클라우드와 데이터를 공유하여 시간 경과에 따른 전기 자동차 배터리 상태, 다양한 주행 조건 전체의 ADAS 센서 정보, 차량 기능 사용과 같은 실제 성능을 문서화할 수 있습니다. 이 데이터는 OEM이 차량 기능을 최적화하고 새로운 문제를 해결하는 데 필요한 시간을 단축할 수 있으며, 특히 ADAS 및 자율 주행과 같은 기술에 도움이 됩니다. 또한 OEM은 특정 차량 모델에서 일반적인 문제를 식별하고 주요 문제가 발생하기 전에 수정 사항을 제공할 수 있습니다.

디지털 트윈 기술 외에도 차량 데이터는 차량, 사람 및 인프라 간에 정보를 공유하여 안전 및 교통 흐름을 향상시킬 수 있기 때문에 V2X 통신에 유용합니다. 차선 이탈 및 차량 속도와 같은 정보를 중앙 컴퓨터에서 다른 차량으로 안전하게 공유하면 충돌 방지 기능을 향상시킬 수 있습니다.

마지막으로, OEM은 새로운 수익원을 창출하는 방법을 지속적으로 모색하고 있습니다. 소프트웨어 정의 차량을 사용하면 OEM이 차량 내의 소프트웨어를 완벽하게 제어할 수 있으므로 사용자 환경을 차별화할 수 있습니다. OEM은 소프트웨어를 통해 활성화할 수 있는 특정 기능 에 대한 구독 모델을 제공할 수 있습니다. 열선내장 시트와 같은 기능은 간단할 수도 있고 고급 주행 안전 기능처럼 더 복잡할 수도 있습니다. 구독이 소비자에게 매력적이지 않을 수 있지만, 소비자가 최신 연식을 구매하도록 요구하는 대신 기존 차량에 대한 소프트웨어 업데이트를 통해 새로운 기능을 추가할 수 있습니다.

OTA 소프트웨어 업데이트 프로세스

OTA(Over-the-air) 또는 FOTA(Firmware-over-the-air) 소프트웨어 업데이트는 반드시 개발, 테스트 후 차량에서 액세스할 수 있는 보안 클라우드 기반 서버에 업로드해야만 실행할 수 있습니다. 또한 차량은 중앙 컴퓨팅 시스템, 영역 컨트롤러 또는 에지 ECU에 업데이트를 다운로드하여 저장할 수 있어야 합니다. 업데이트가 효과를 발휘하려면 보통 ECU를 다시 시작해야 하기 때문에 차량이 안전한 상태일 때 업데이트 프로세스를 진행해야 합니다.

업데이트 가능한 경우 운전자에게 이용 가능한 업데이트가 있다고 알리고, 차량을 안전하게 주차한 다음에 운전자가 업데이트 시작을 확인하도록 할 수 있습니다. 아니면 시스템이 차량 사용 시간을 추적하여 사용자가 개입하지 않고도 소프트웨어를 업데이트할 가장 적합한 시간을 추측할 수 있습니다. 이 시간에는 일시적으로 차량을 가동하지 못할 수 있으므로, 업데이트를 효율적으로 완료하여 다운타임을 최소화해야 합니다. ECU는 업데이트하는 동안 전원의 공급 상태를 유지해야 하고, 차량의 배터리 용량도 고려해야 합니다. ECU는 위험을 완화하기 위해 현재 소프트

웨어 버전과 새로운 업데이트를 모두 메모리에 저장하여 다음에 시동할 때 업데이트된 버전으로 전환하도록 설계되어 있습니다. 업데이트가 실패하더라도 시스템이 이전 소프트웨어 버전으로 되돌릴 수 있으므로 차량이 계속 기능하도록 보장됩니다.

차량 기능이 여러 ECU에 분산된 경우, 신중하게 계획한 업데이트 캠페인에 따라 업데이트를 조율해야 합니다. 이러한 캠페인에는 영향을 받는 모든 ECU에 업데이트 패키지를 배포하는 과정을 포함해 시스템 전체의 호환성과 성능을 보장해야 합니다.

소프트웨어 정의 차량 및 영역 아키텍처 접근 방식의 변화

각 자동차 제조업체는 소프트웨어 정의 차량을 달성하기 위한 고유한 접근 방식을 가지고 있습니다. 이전 세대 차량 플랫폼의 전통에 따라 많은 OEM은 중앙 집중식 소프트웨어 접근 방식을 더 잘 충족하는 전기 및 전자 영역 아키텍처로 점진적으로 전환하게 될 것입니다.

대부분의 OEM은 영역 아키텍처를 개발하고 있지만, **그림 5**에서 볼 수 있듯이 차량 기능을 제어할 소프트웨어가 어디에 있는지 결정할 때는 다양한 접근 방식이 있습니다.

중앙 컴퓨터, 중앙 컴퓨터와 영역 제어 모듈 간 공유 또는 일부 도메인 컨트롤러 및 영역 제어 모듈에 배포하는 등 소프트웨어 제어를 중앙 집중화할 때는 세 가지 옵션이 있습니다. 일부 OEM은 ADAS 및 차량 내 인포테인먼트와 같은 고성능 컴퓨팅 도메인을 중앙 집중화하고 다른 도메인에 대한 애플리케이션 처리 기능을 추가하고 있습니다. ADAS 및 차량 내 인포테인먼트 도메인 외에도 영역 제어 모 듈 또는 에지 ECU에서 실시간 제어가 구현됩니다.

중앙 집중식 컴퓨팅 접근 방식은 OEM 관점에서 가장 매력적일 수 있으며, 이는 단일 컴퓨터가 모든 차량 기능을 제어하기 때문입니다. 통신 링크가 실패할 경우 실시간 제어 루프 지연(액티브 서스펜션, 윈도우 안티 핀치) 및 기능 안전과 관련하여 추가적인 문제가 있을 수 있습니다.

분산 컴퓨팅 접근 방식은 소프트웨어를 중앙 집중화하고, 영역 제어 모듈 또는 별도의 도메인 컨트롤러에서 일부 애플리케이션과 실시간 제어 소프트웨어를 유지하는 방향으로 보다 점진적인 단계를 거칩니다. 모든 아키텍처에서 영역 제어 모듈 요구 사항은 OEM에 따라 동일한 차량 내에서도 다릅니다. 한 영역은 차체의 실시간 제어, 난방, 환기 및 에어컨, 샤시 기능을 처리할 수 있으며, 다른 영역은 추가적인 차체, 조명 및 차량 제어 장치 애플리케이션 소프트웨어를 처리할 수 있습니다. 궁극적으로 OEM은 하드웨어 및 기계적 작동 제어 지연 시간, 차량 내 네트워크 기능, 기능 안전, 보안, 선택한 아키텍처와 특정 구역 제어 모듈 요구 사항에 맞는 소프트웨어 구성 방법의 균형을 맞춰야 합니다.

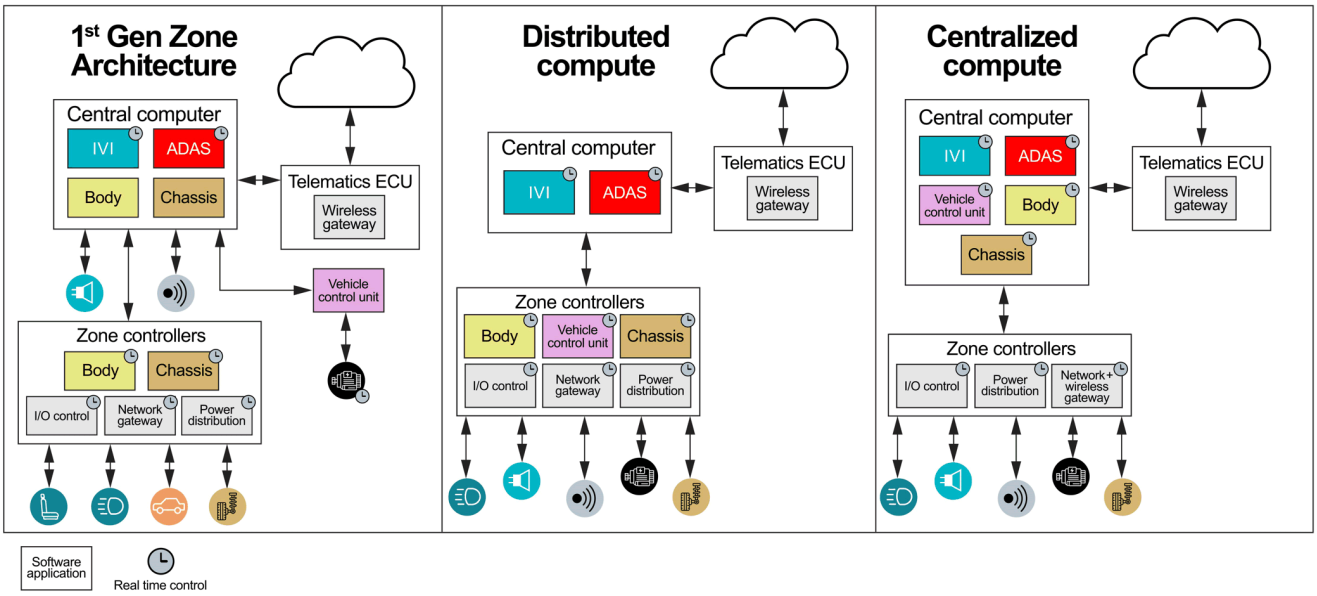

그림 5. *차량 아키텍처 유형 비교.*

마무리

소프트웨어 정의 차량은 자동차 제조업체가 새로운 차량과 기능 개발에 관련된 시간과 비용을 줄이고, 차량의 수명 전반에 걸쳐 운전자 경험을 지속적으로 개선하며, 새로운 수익원을 창출할 수 있는 새로운 기회를 만들고 있습니다. 여러 가지 접근 방식이 있지만 차량 소프트웨어의 중앙 집중화와 소프트웨어에서 차량 하드웨어를 추상화하는 것이 가장 우선순위가 될 것입니다. 전반적으로, OEM은 영역 아키텍처와 소프트웨어 정의 차량을 통해 더 스마트하고 안전하며 에너지 효율적인 차량의 개발을 가속화할 것입니다.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), 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 will 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 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.

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

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265