CONTENT OF SOFTWARE LAYERS OF AUTOSAR ARCHITECTURE

MCAL

Microcontroller Drivers

组成/分类：

- General Purpose Timer(GPT) Driver
- Watchdog Driver
- MCU Driver
- Core Test

其中，有内部外设驱动，比如：Watchdog、General Purpose Timer（GPT）；

直接访问MCU的函数，比如：Core test。

Memory Drivers

组成/分类：

- Internal EEPROM Driver
- Internal Flash Driver
- RAM Test
- Flash Test

主要是片上存储设备驱动，比如：Internal Flash、Internal EEPROM；

以及内存映射的外部存储驱动设备，比如：External Falsh。

Crypto Drivers

组成/分类：

- Crypto Driver

片上加密设备的驱动，比如：SHE、HSM。

Communication Drivers

组成/分类：

- Ethernet Driver
- FlexRay Driver
- CAN Driver
- LIN Driver
- SPI Handler

主要是ECU板上驱动，比如：SPI；

和汽车通信的驱动，比如：CAN；

以及OSI-Layer：数据链路层的一部分。

Wireless Communication Drivers

组成/分类：

- Wireless Ethernet Driver

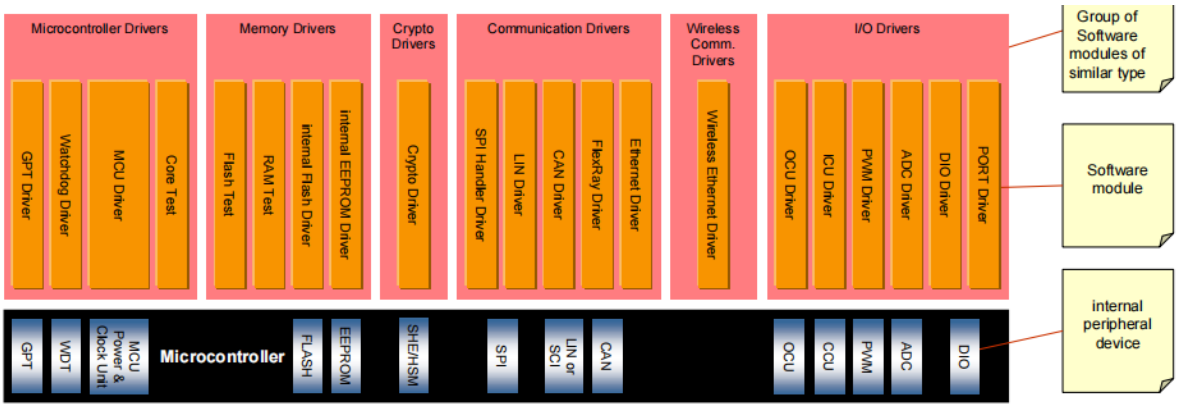
主要是无线网络系统的驱动（车载或非车载通信）。

I/O Driver

组成/分类：

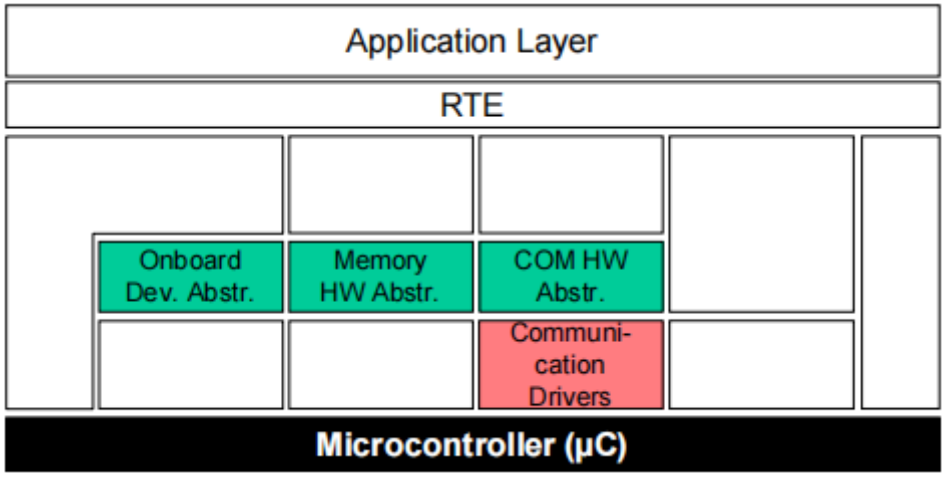
- PORT Driver
- DIO Driver
- ADC Driver
- PWM Driver
- ICU Driver
- OCU Driver

主要是模拟和数字I/O的驱动，比如：ADC、PWM、DIO。



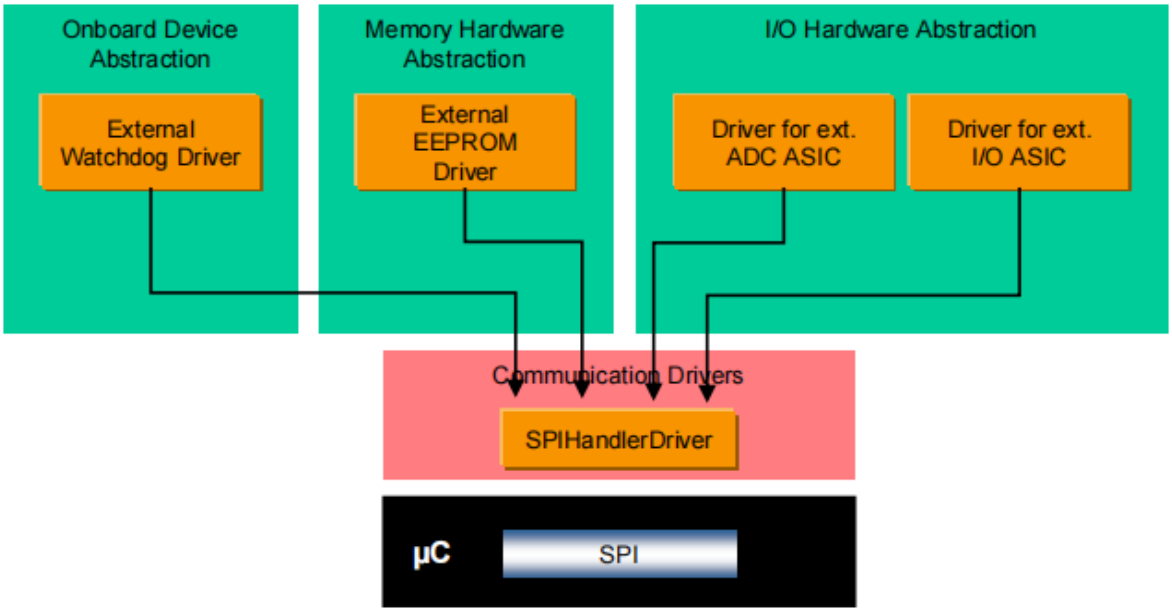
SPIHandlerDriver of MCAL

SPIHandlerDriver允许多个客户端对一个或多个SPI总线进行并发访问。



为了抽象出专用于芯片选择的SPI微控制器引脚的特性/功能，这些特性/功能应该直接由SPIHandlerDriver直接处理。这意味着，这些CS引脚在DIO驱动中不可用。

MCAL的SPIHandlerDriver可被ECU抽象层的多种/多个功能模块调用，例如：



Complex Drivers

一个Complex Driver是一个在基础软件栈中实现非标准功能的模块。

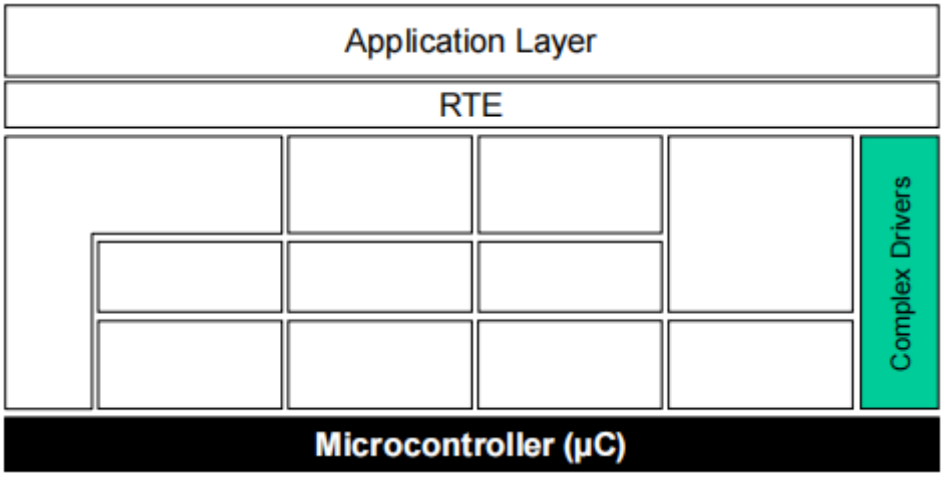
示例：使用特定的中断and/or复杂的MCU外设（比如：PCP、TPU）直接访问MCU，从而实现复杂传感器检测和执行器控制，比如：Injection control、Electric valve control、Incremental position detection。

主要作用：为操作复杂传感器和执行器实现特定的功能和时间需求。

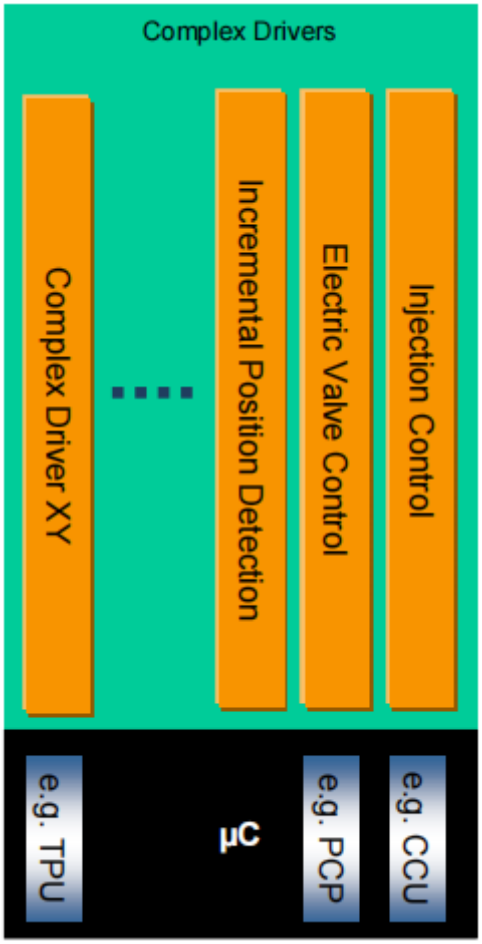
实现依赖：高度依赖于MCU、ECU和应用软件。

为上层SW-Cs根据AUTOSAR（AUTOSAR interface）提供/实现特定的功能/接口。

对于下层，访问标准化接口时受限。



示例：



ECU Abstraction Layer

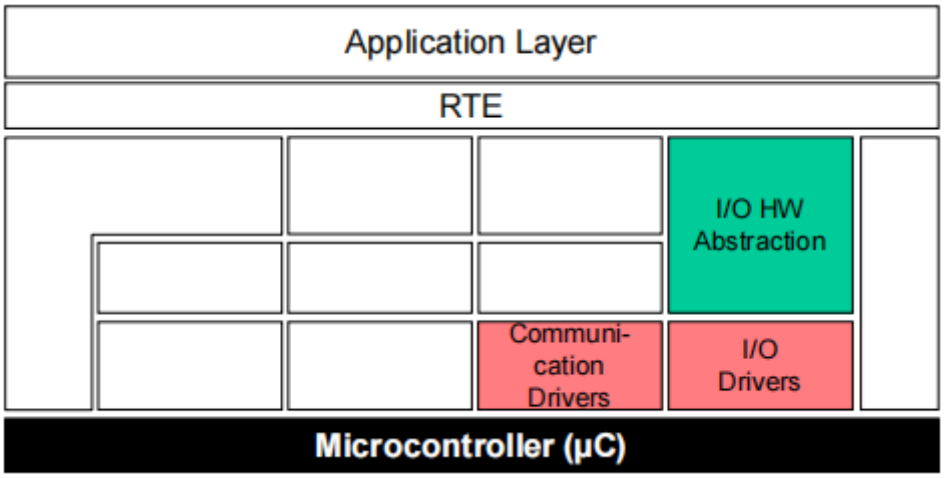
I/O Hardware Abstraction

I/O硬件抽象是一组抽象了外围I/O设备（片上或板上）和ECU硬件布局（比如：MCU引脚了解和电平信号倒置）的位置的模块。I/O硬件抽象并不会抽象传感器或者执行器。

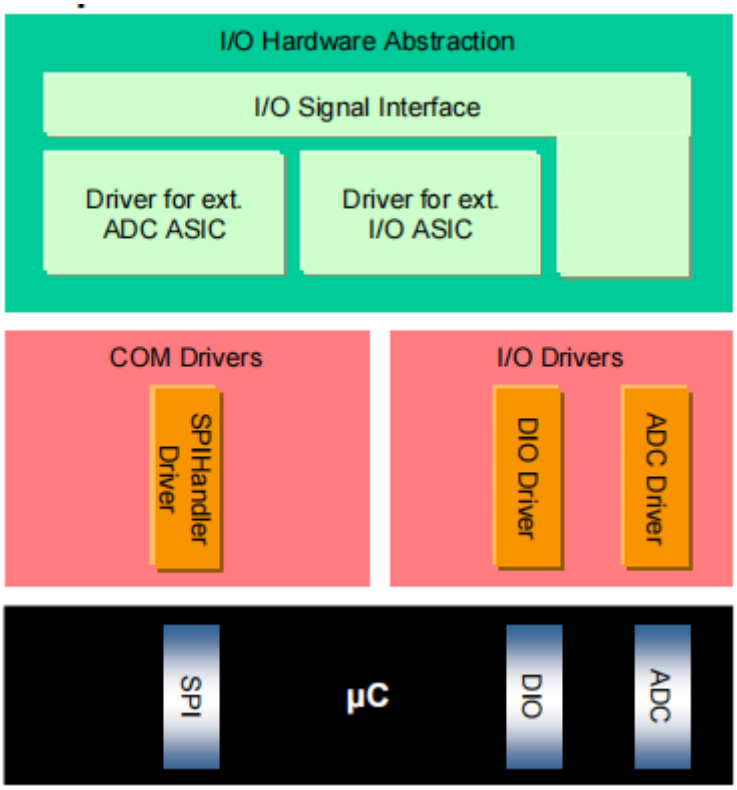
不同的I/O设备可能通过一个I/O信号接口访问。

主要作用：

- 表示I/O信号，当I/O连接到ECU硬件时（比如：电流、电压、频率）；
- 从更高软件层中隐藏ECU硬件和布局的属性。



示例：



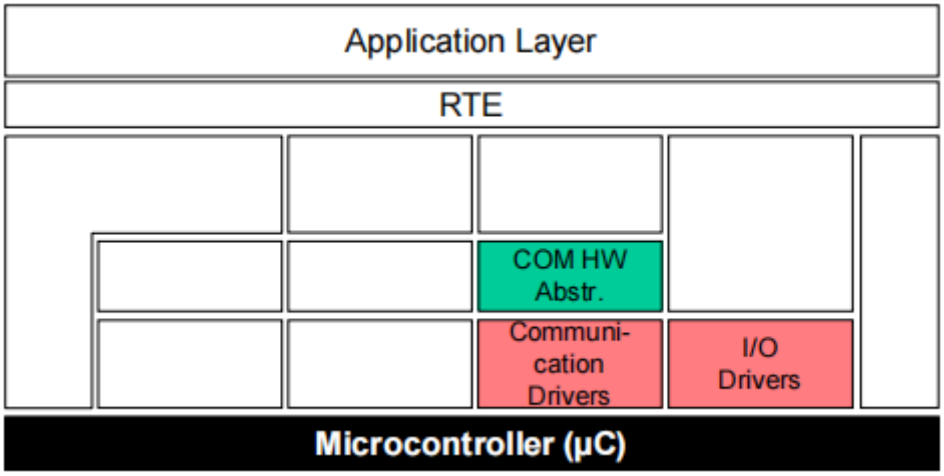
Communication Hardware Abstraction

通信硬件抽象是一组抽象了通信控制器和ECU硬件布局的位置的模块。对于所有通信系统，一个特定的通信硬件抽象是必要的。

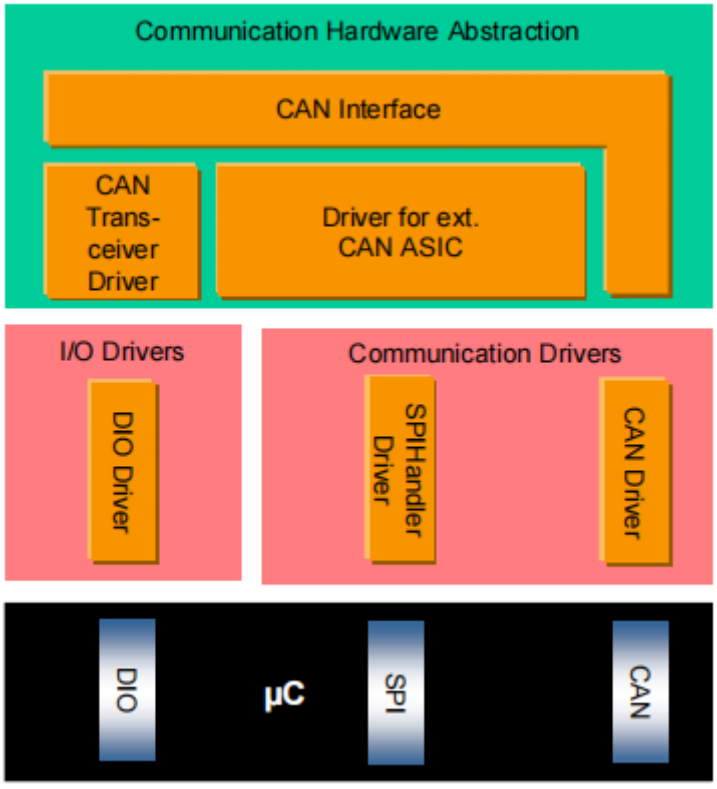
主要作用：提供一个相同的机制来访问总线通道，而不必理会总线的位置（片上或者板上）。

实现依赖：独立于MCU，但依赖于ECU的其他硬件（比如板上CAN控制器）和其他外部设备。

其上层依赖于总线，但独立于MCU和ECU硬件。



示例：一个ECU有一个带着两个内部CAN通道的MCU和一个带着4个CAN控制器的板上ASIC(Application Specific Integrated Circuit 特定用途集成电路)。这个CAN-ASIC通过SPI与MCU相连。则可通过总线特定接口（CAN Interface）（在ECU Abstraction Layer上）访问通信驱动（如CAN Driver）而不必理会是片上CAN还是板上CAN，也不用关注是通过MCU的CAN接口还是SPI接口连接的。



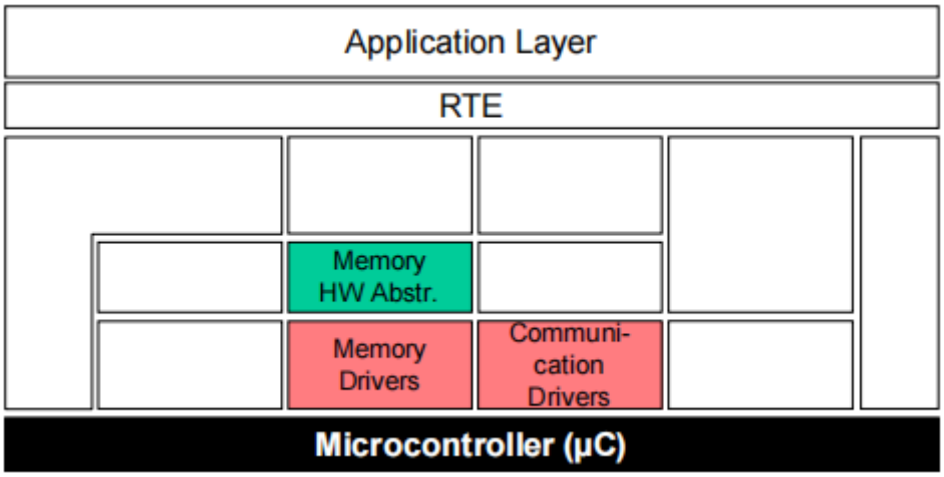
Memory Hardware Abstraction

存储硬件抽象是一组抽象了外围存储器设备（片上或板上）和ECU硬件布局的位置的模块。

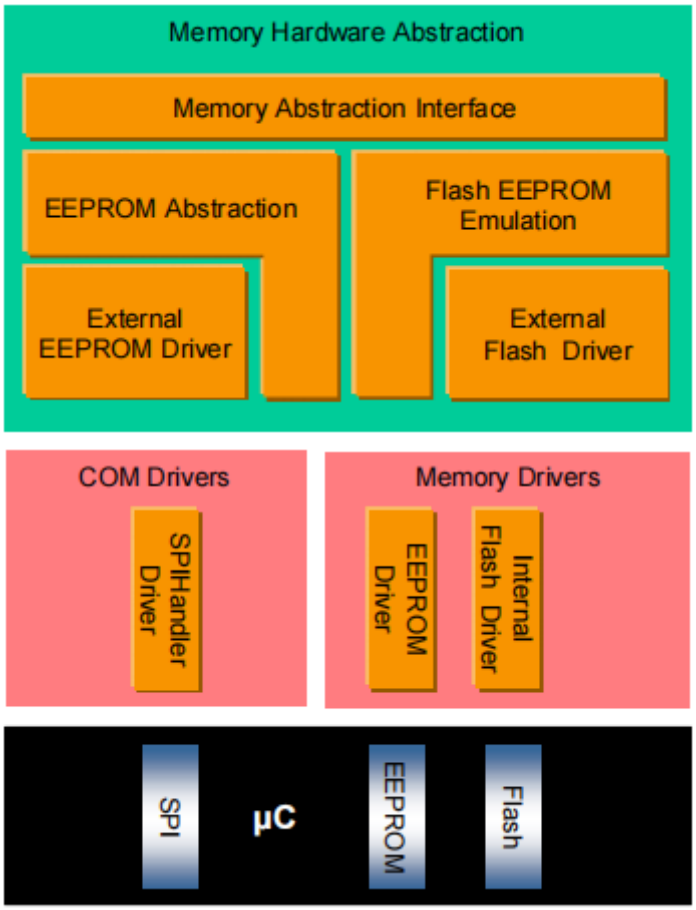
主要作用：提供一个相同的机制访问内部（片上）和外部（板上）存储设备和存储硬件类型（比如：EEPROM、Flash）。

实现依赖：独立于MCU，但依赖于外部设备。

其上层独立于MCU、ECU硬件和存储设备。



示例：片上EEPROM和外部EEPROM都可以通过相同的机制访问。通过存储器特定的抽象/仿真模块（比如：EEPROM Abstraction）可以访问存储器驱动（On MCAL）。通过在Flash硬件单元上模拟EEPROM抽象，启用了通过内存抽象接口对这两种类型硬件的通用访问。



Onboard Device Abstraction

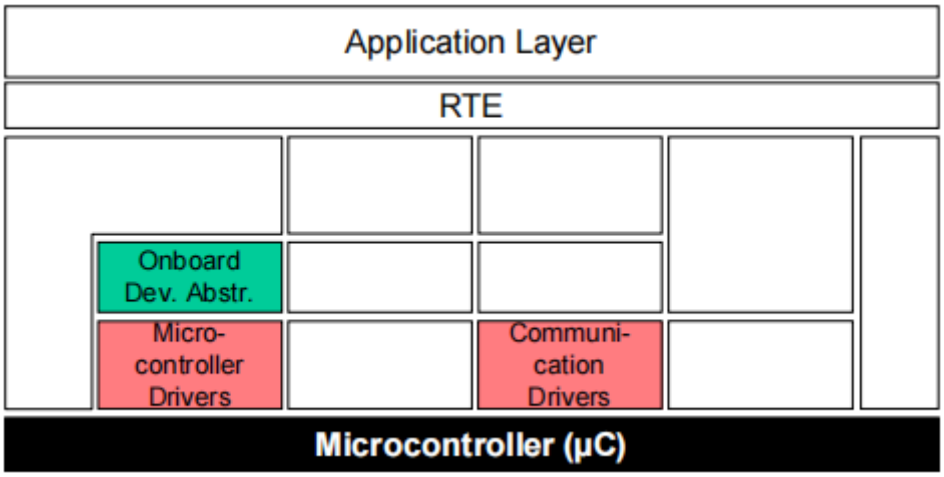
板上设备驱动包含了不能视作传感器和执行器（比如：内部或外部watchdogs）的ECU板上设备的驱动。这些驱动通过MCAL层访问ECU板上设备。

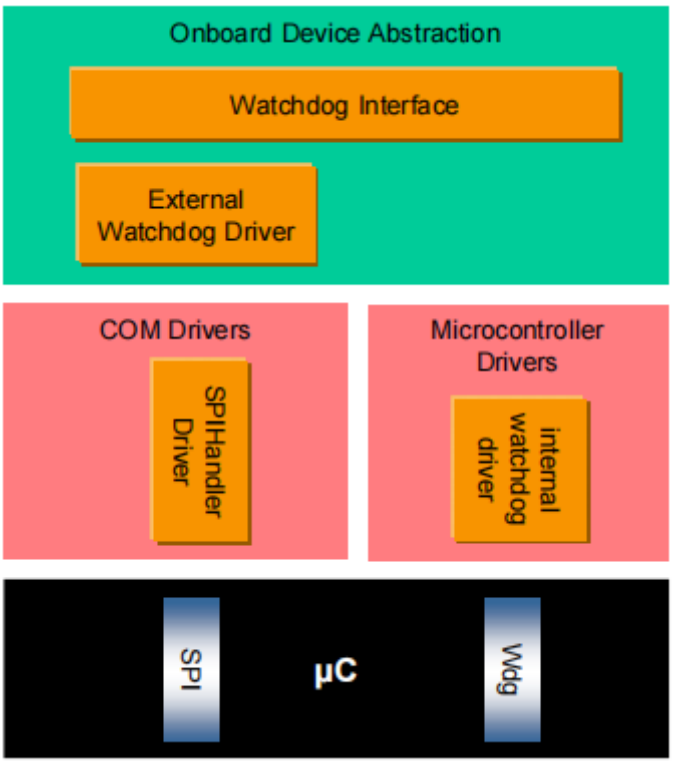
主要作用：抽象ECU特定的板上设备。

实现依赖：独立于MCU，但依赖于外部设备。

其上层独立于MCU，但部分依赖于ECU硬件。

示例：





Crypto Hardware Abstraction

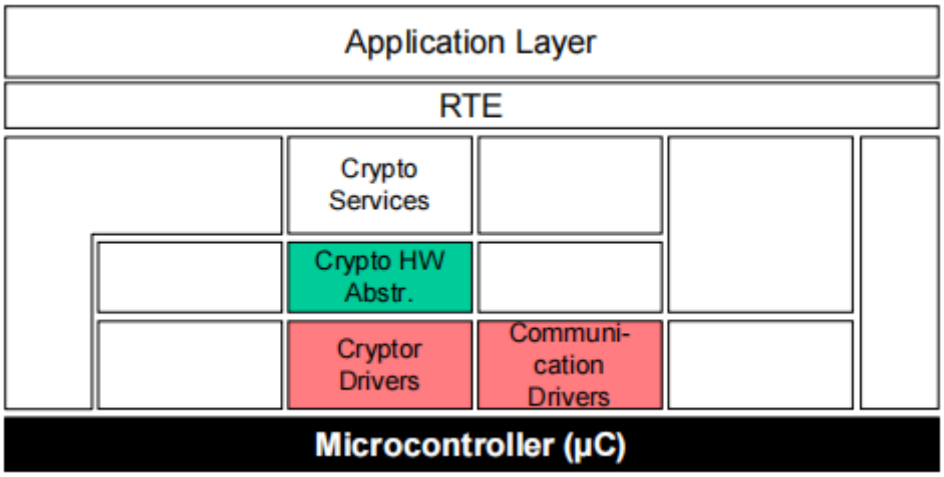
加密硬件抽象是一组抽象了密码原语（内部/外部硬件，或者基于软件的密码原语）的位置。

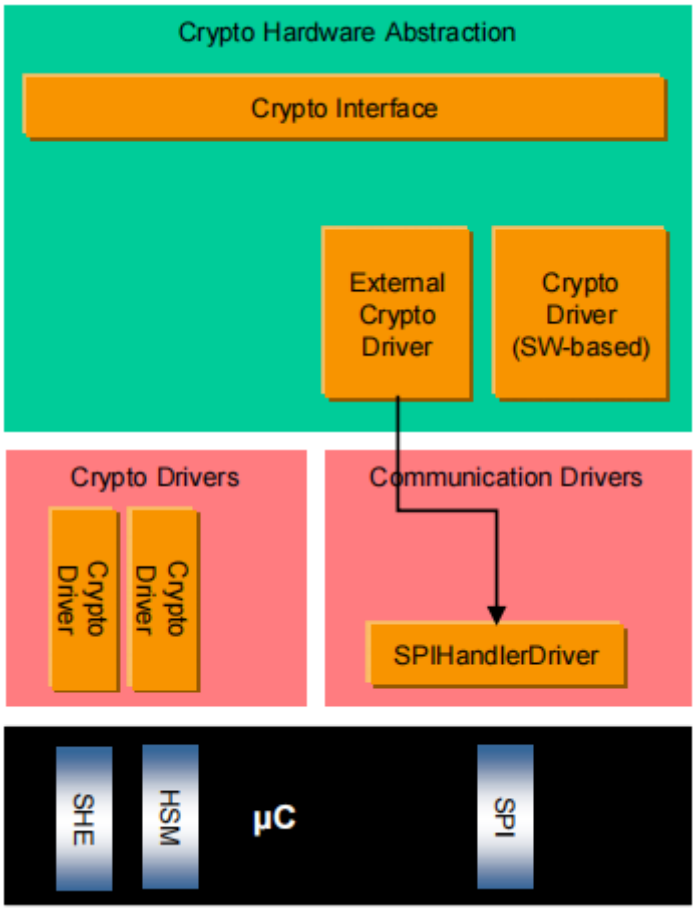
主要作用：提供相同的机制访问内部（片上）/外部（板上）设备，或者软件加密设备。

实现依赖：独立于MCU。

其上层独立于MCU、ECU硬件和加密设备。

示例：AES原语在SHE中实现或者作为软件库提供给外部。





Services Layer

Crypto Services

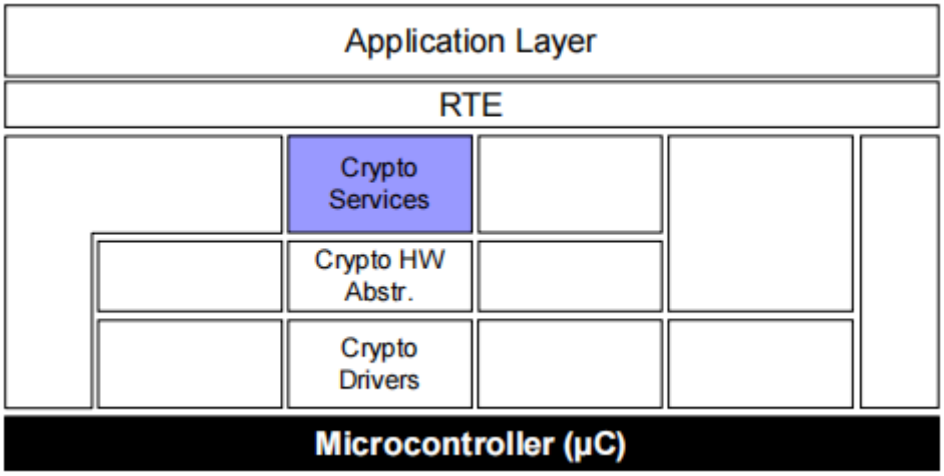
加密服务主要包含三个模块：

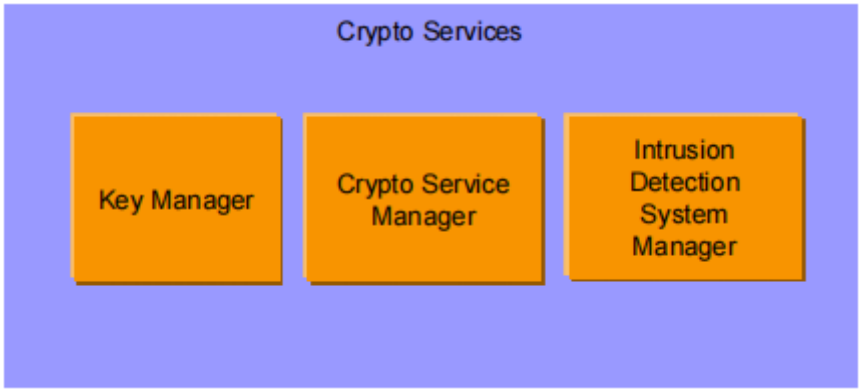
- Crypto Service Manager：负责管理加密作业。
- Key Manager（密钥管理器）：与密钥配置主服务器（在NVM或者加密驱动中）进行交互，并管理证书链的存储和验证。
- Intrusion Detection System Manager（入侵检测系统管理器）：负责处理BSW模块或者SW-C报告的安全事务。

主要作用：以统一方式为应用软件提供加密原语、IDS服务和秘钥存储功能/服务；抽象硬件设备和属性。

实现依赖：独立于MCU和ECU硬件，高度可配置。

其上层独立于MCU和ECU硬件；根据AUTOSAR(AUTOSAR Interface)指定和实现。





Communication Services

General

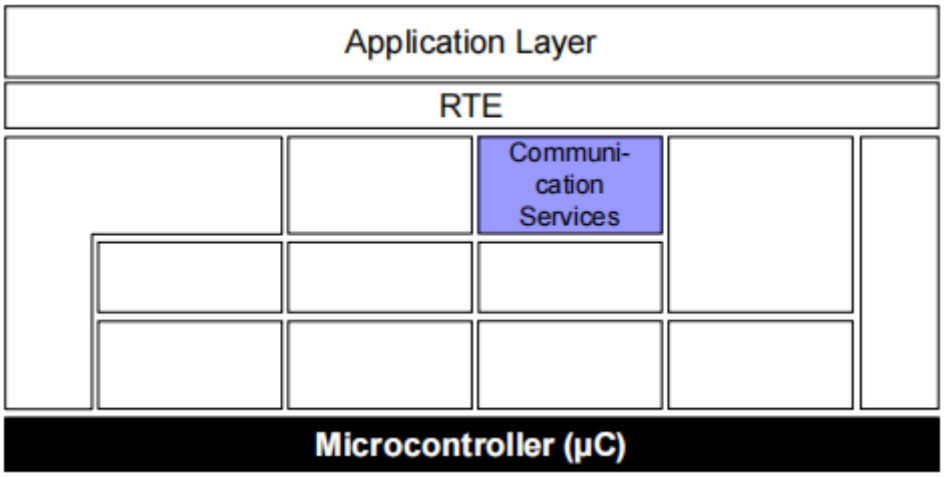
通信服务是一组车载网络通信模块（CAN、LIN、FlexRay、Ethernet），通过Communication Hardware Abstraction与Communication Drivers相连。

主要作用：

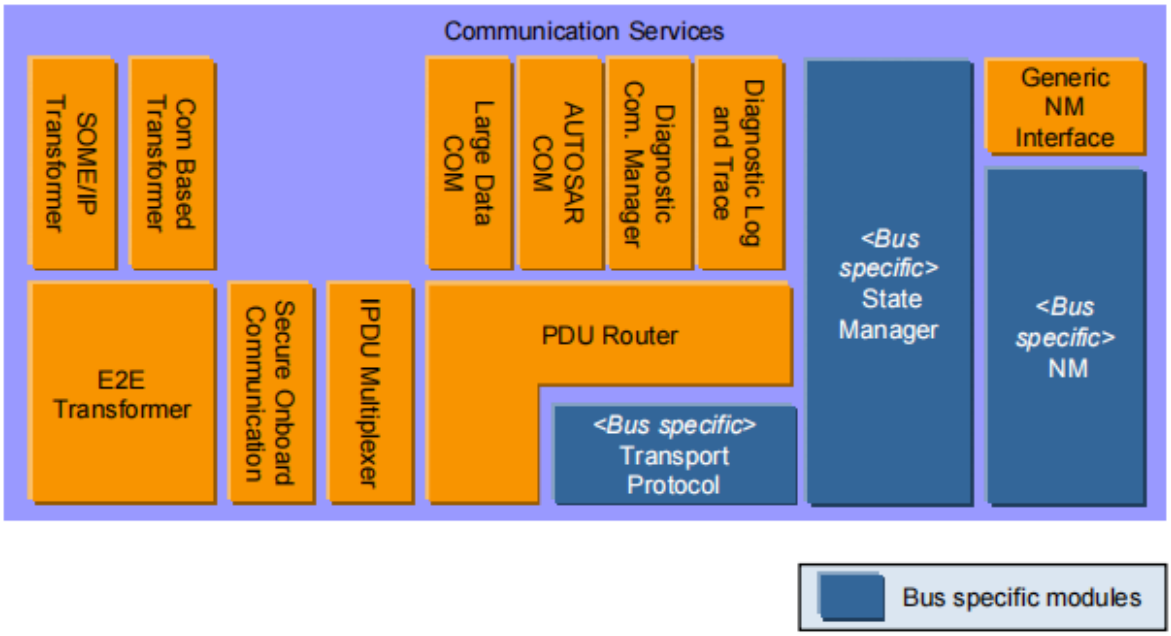
- 为车载网络通信提供统一接口；
- 为网络管理提供统一服务；
- 为车载网络诊断通信提供统一接口；
- 为上层应用软件隐藏协议和报文属性。

实现依赖：独立于MCU和ECU硬件，但部分依赖于总线类型。

其上层独立于MCU、ECU硬件和总线类型。



服务类型：



Communication Stack - CAN

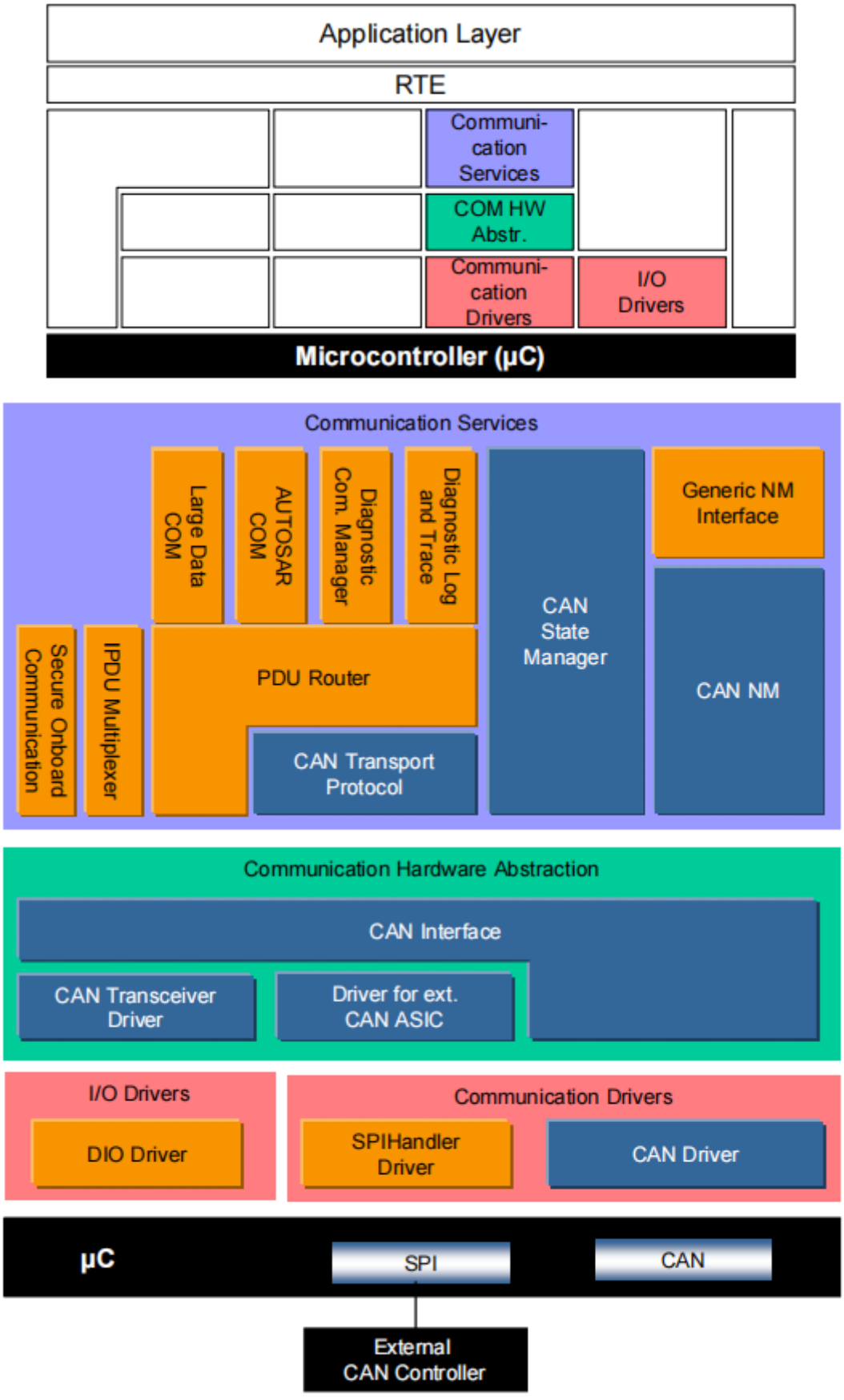
CAN Communication Services是一组为带CAN通信系统的车载网络通信提供的模块。

主要作用：为CAN网络提供统一接口；为上层应用软件隐藏协议和报文属性。

CAN Communication Stack支持：Classic CAN communication(CAN 2.0); CAN FD communication, if supported by hardware.

属性：

- 实现依赖：独立于MCU和ECU硬件，但部分依赖于CAN。
- 对于所有车载网络系统来说，AUTOSAR COM、通用NM(Network Management) 接口和Diagnostic Communication Manager是相同的；且对于每个ECU来说，他们作为一个实例。
- 通用NM接口只包含一个调度程序，没有更多的功能。对于网关ECU来说，它还可以包括NM协调器功能， 该协调器允许同步多个不同的网络，以同步唤醒或关闭他们。
- CAN NM是CAN网络专用的，且为每个CAN车载网络系统分别实例化。
- 通信系统特定的CAN State Manager处理与通信系统相关的启动和关闭功能。此外，它还控制着COM的不同选项来发送PDUs和监控信号超时。



Communication Stack Extension - TTCAN

TTCAN Communication Services是普通CAN接口和CAN驱动模块的可选扩展，用于带有通信系统TTCAN的车载网络通信。

主要作用：为TTCAN网络提供一个统一接口；为上层应用软件隐藏协议和报文属性。

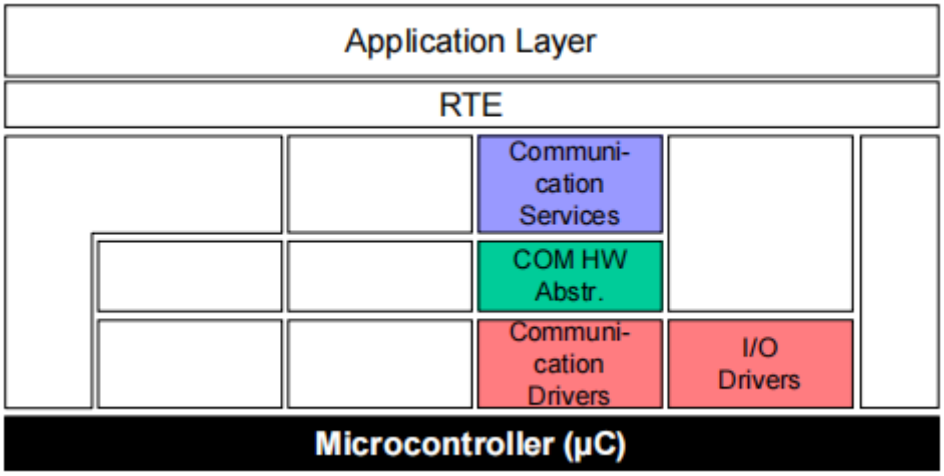
注意：带TTCAN的CAN接口可以作为普通CAN和CAN Driver TTCAN。

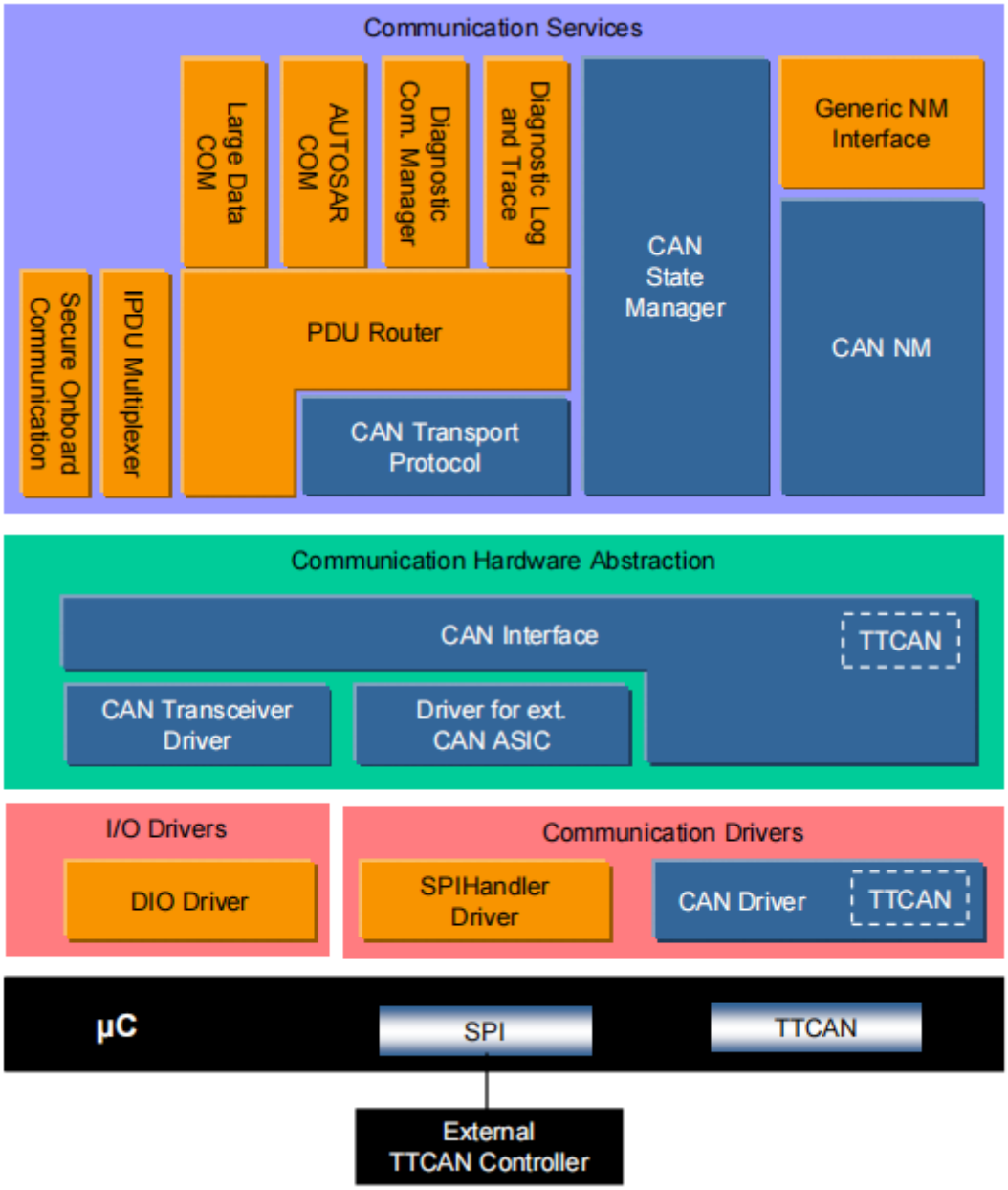
属性：

TTCAN是CAN的一个绝对超集，即，一个支持TTCAN的CAN栈可以作为一个CAN和一个TTCAN总线。

CanIf和CanDrv是唯一需要扩展来服务TTCAN通信的模块。

通信栈CAN的属性也适用于带TTCAN功能的CAN。





Communication Stack Extension - J1939

J1939 Communication Services扩展了普通CAN通信栈，应用于重型汽车中的车载网络通信。

主要作用：提供J1939需要的协议服务；对不需要的应用程序隐藏了协议和报文属性。

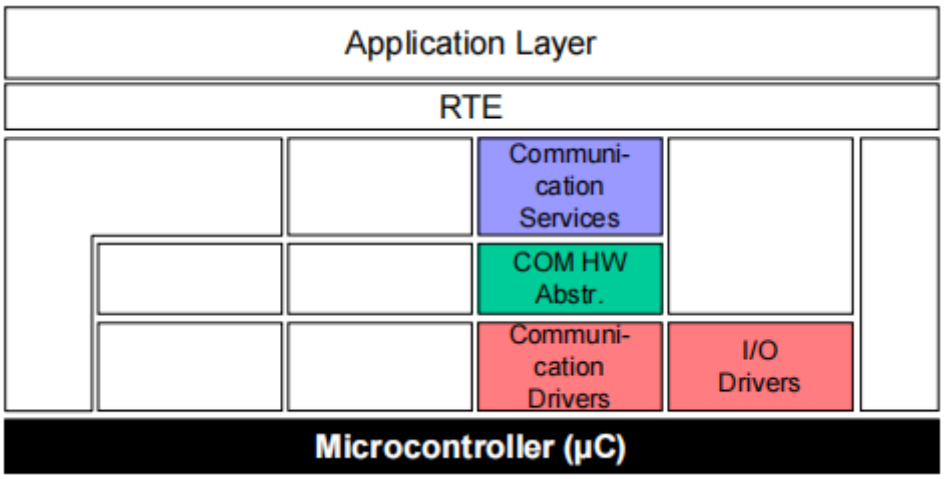
注意：在CAN栈中有两个传输协议模块：Can Tp和J1939Tp，两者可在不同通道上交替或并行使用。用途分别为：

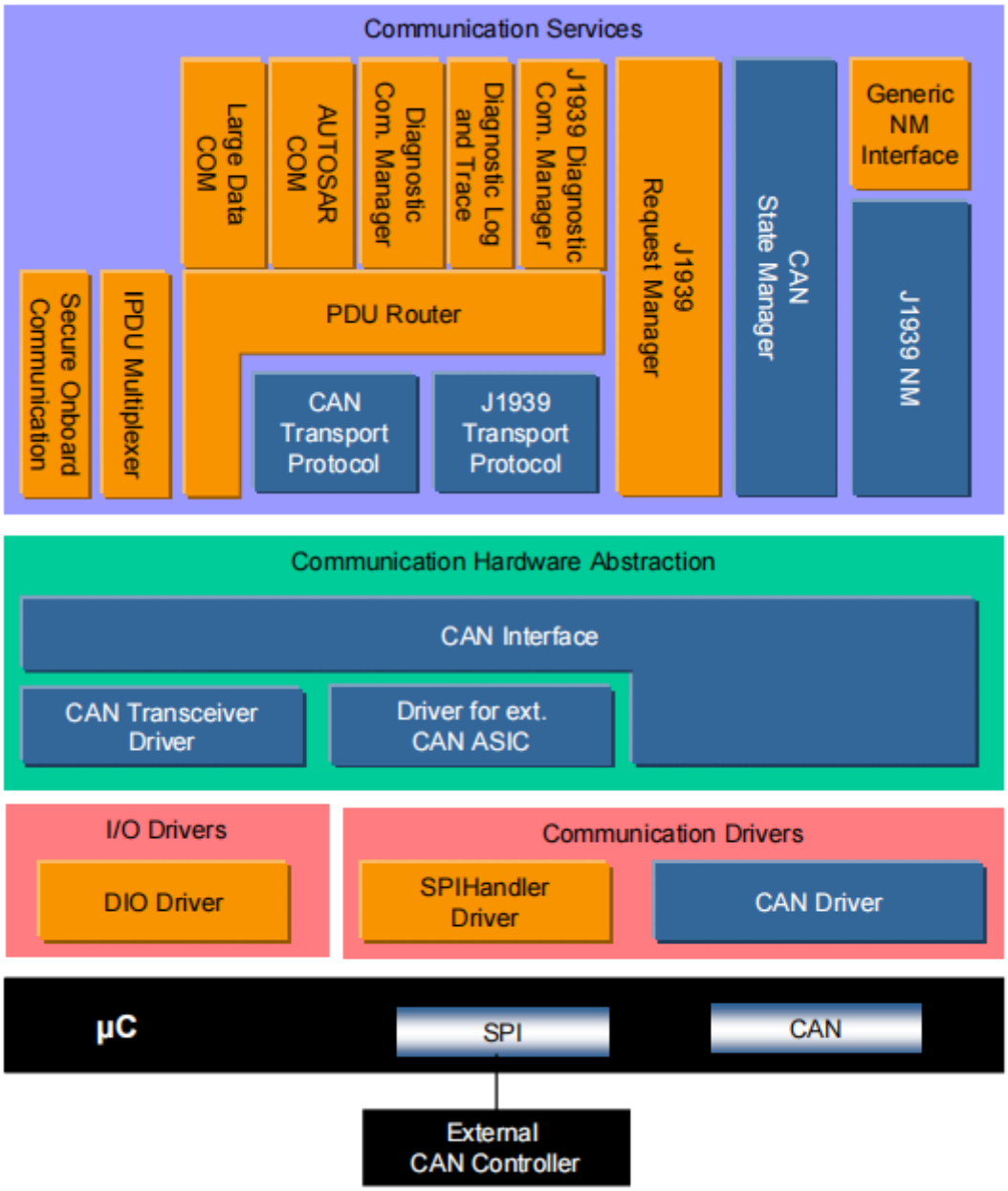
CanTp: ISO Diagnostics(DCM), large PDU transporton standard CAN bus;

J1939Tp: J1939 Diagnostics, large PDU transport onJ1939 driven CAN bus.

属性：

- 实现依赖：独立于MCU和ECU硬件，但基于CAN。
- 对于所有车载网络系统来说，AUTOSAR COM、通用NM(Network Management) 接口和Diagnostic Communication Manager是相同的；且对于每个ECU来说，他们作为一个实例。
- 支持在配置时使用未知的动态标志符。
- J1939网络管理处理每个ECU的唯一地址分配，但不支持休眠/唤醒处理和相关概念，比如部分网络（partial networking）。
- 提供J1939诊断和请求处理。





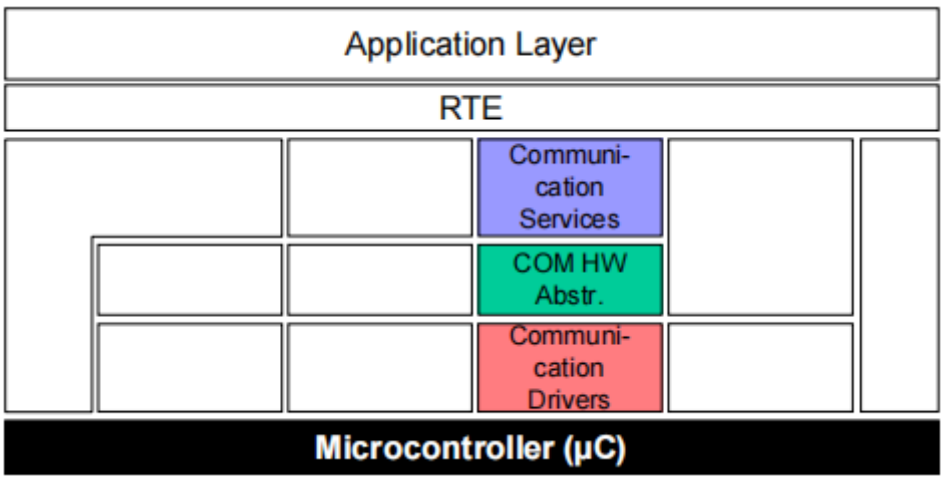
Communication Stack - LIN

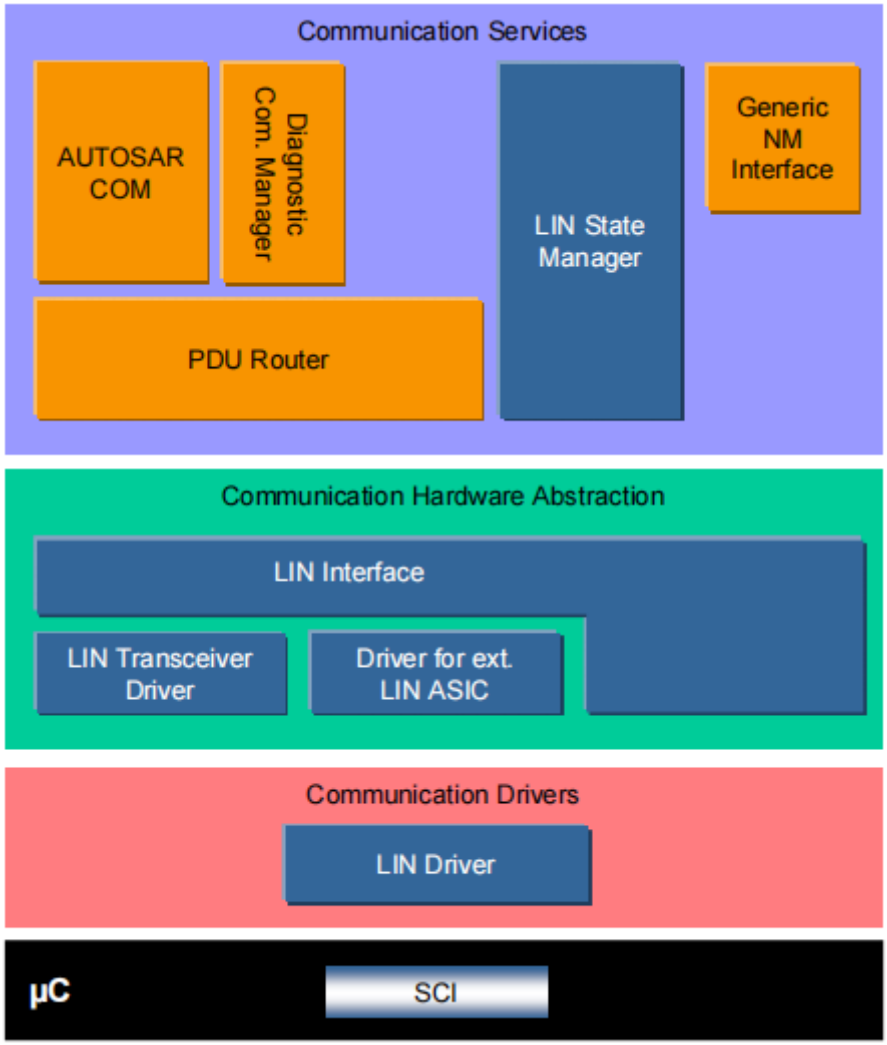
LIN Communication Services 是一组为带通信系统LIN的车载网络提供的模块。

主要作用：为LIN网络提供统一接口；为上层应用软件隐藏协议和报文属性。

属性：

注意：





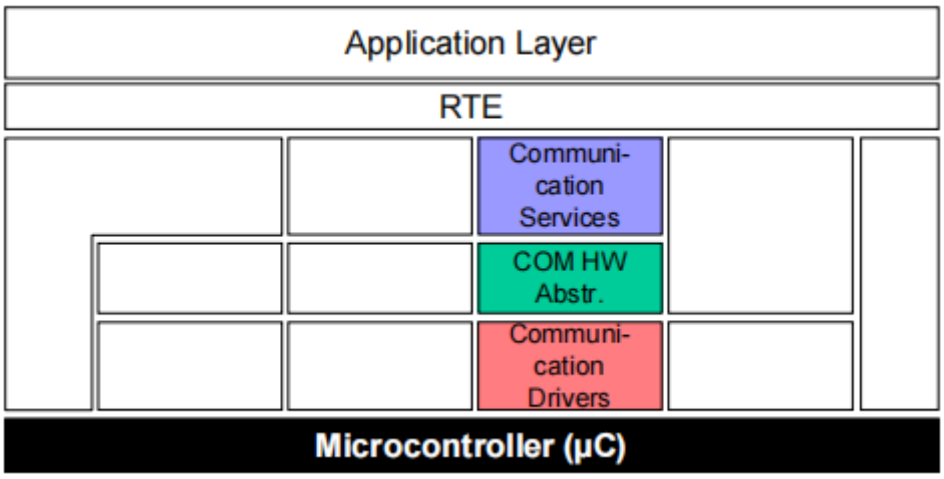
Communication Stack - FlexRay

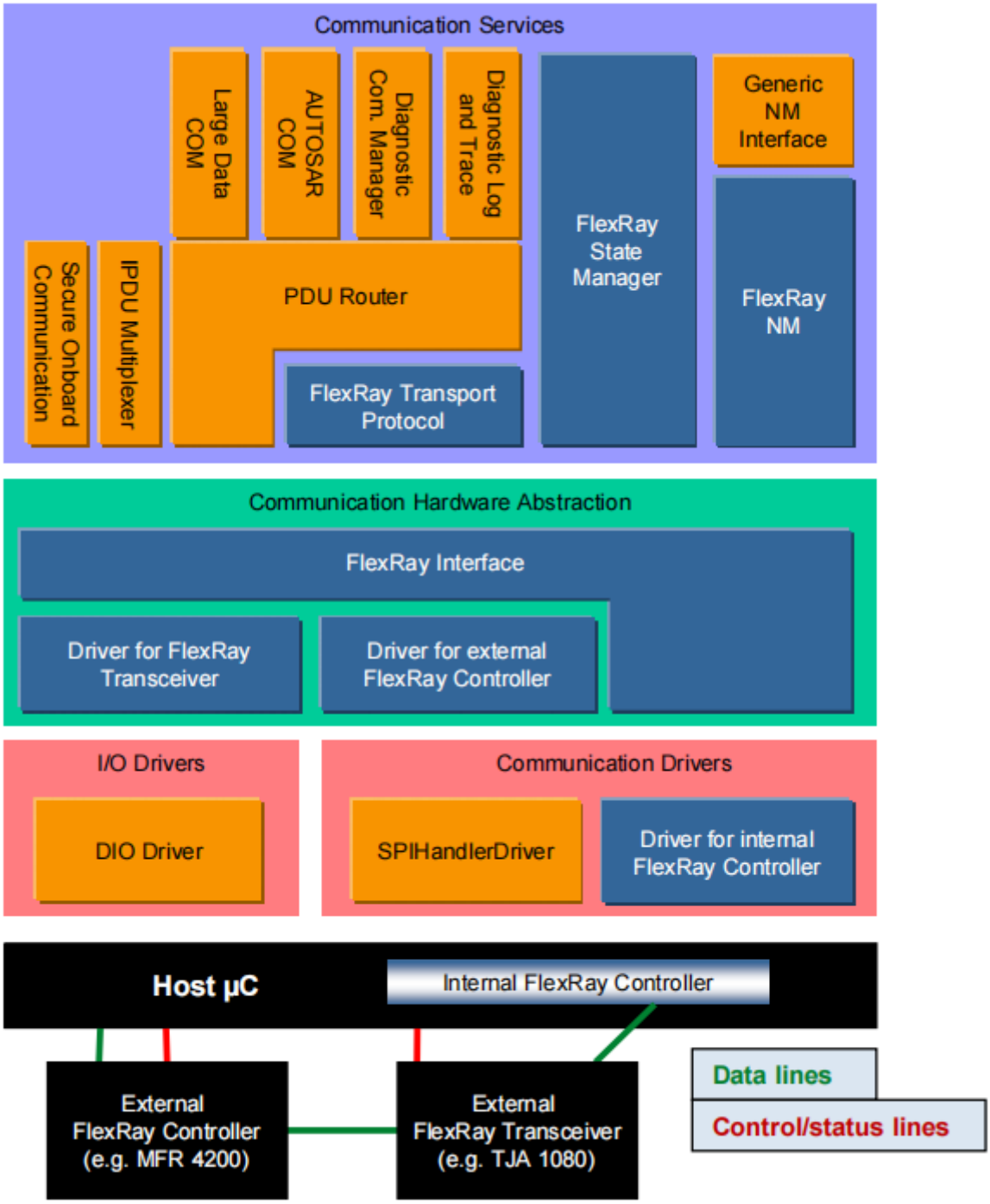
FlexRay Communication Services是一组为带通信系统FlexRay的车载网络提供的模块。

主要作用：为FlexRay网络提供统一接口；为上层应用软件隐藏协议和报文属性。

属性：

注意：





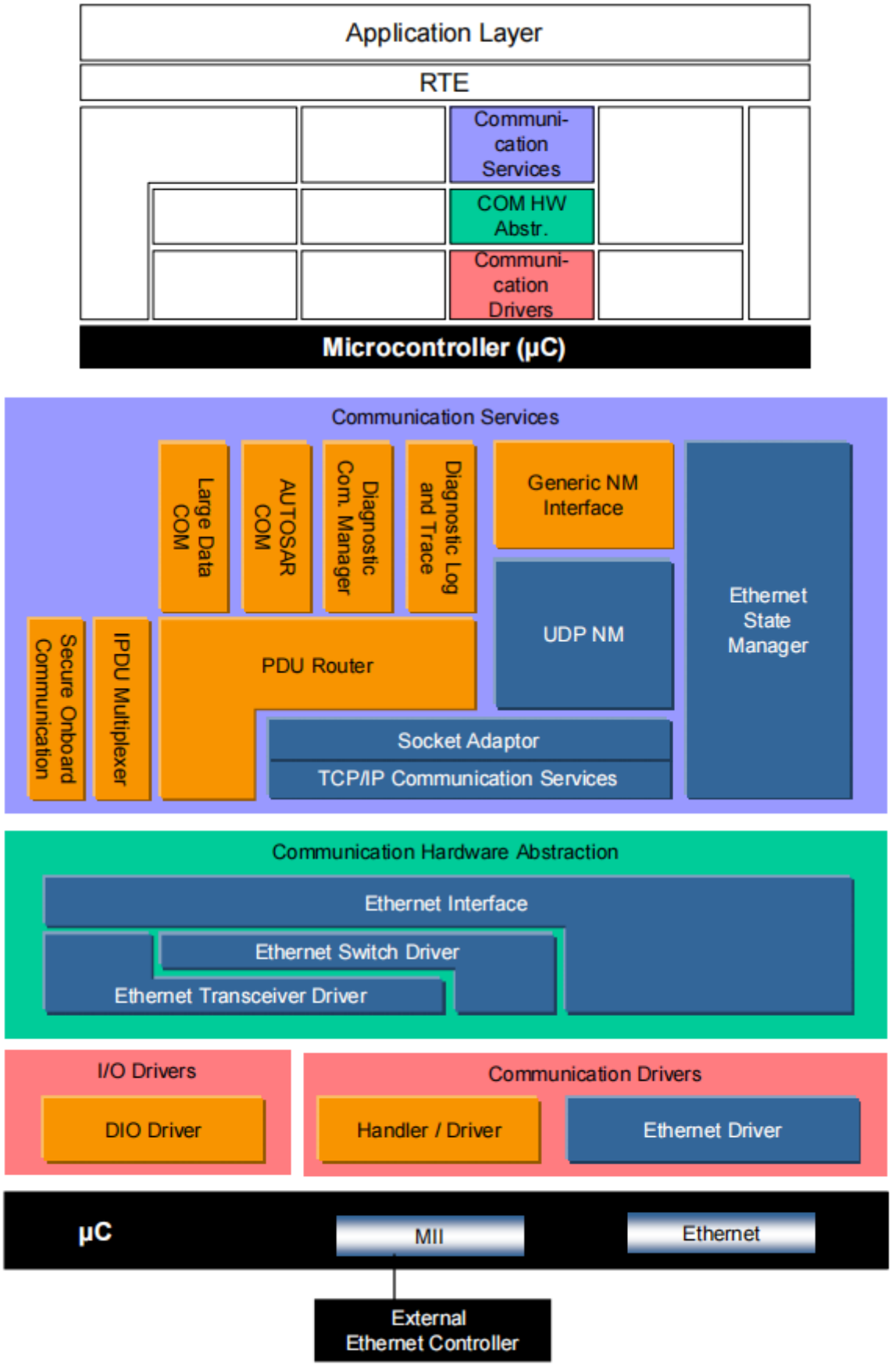
Communication Stack - TCP/IP

TCP/IP Communication Services是一组为带通信系统TCP/IP的车载网络通信提供的模块。

主要作用：为TCP/IP网络提供统一接口；为上层应用软件隐藏协议和报文属性。

属性：

- TcpIp模块实现了TCP/IP协议族的主要协议（TCP、UDP、IPv4、IPv6、ARP、ICMP、DHCP），并通过以太网提供了动态、基于socket的通信；
- Socket Adaptor module（SoAd）是TcpIp模块的单独的上层模块。



Communication Stack - General

Off-board Communication Stack - Vehicle-2-X

Memory Services

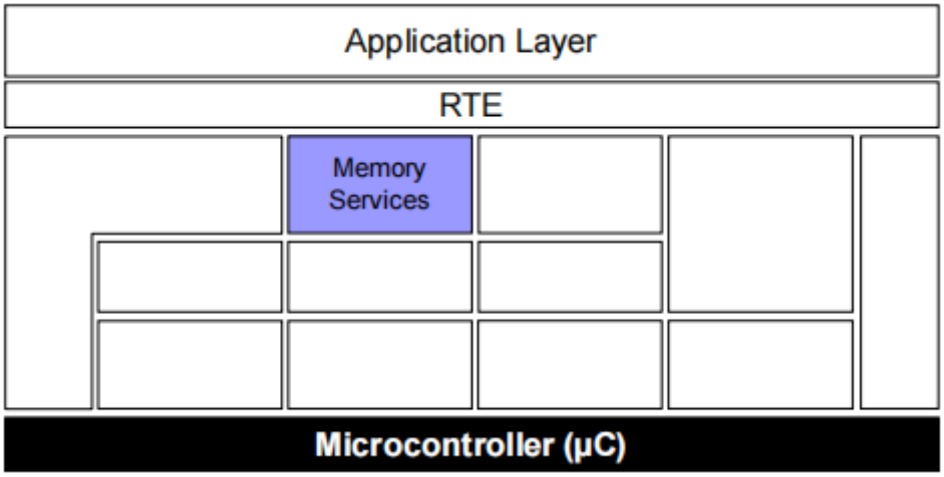
Memory Services由一个模块组成：NVRAM Manager，负责非易失性数据的管理（读写不同的存储设备）。

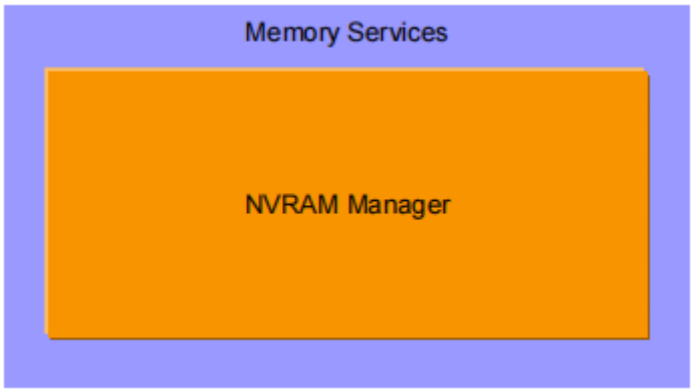
主要作用：

- 以一个统一的方式为应用软件提供非易失性数据；
- 抽象出存储器地址和其属性；
- 为非易失性数据提供机制，比如：保存、加载、检验保护、验证、可靠存储等。

实现依赖：独立于MCU和ECU硬件；高度可配置性。

其上层独立于MCU和ECU硬件，根据AUTOSAR(AUTOSAR interface)指定和实现。





System Services

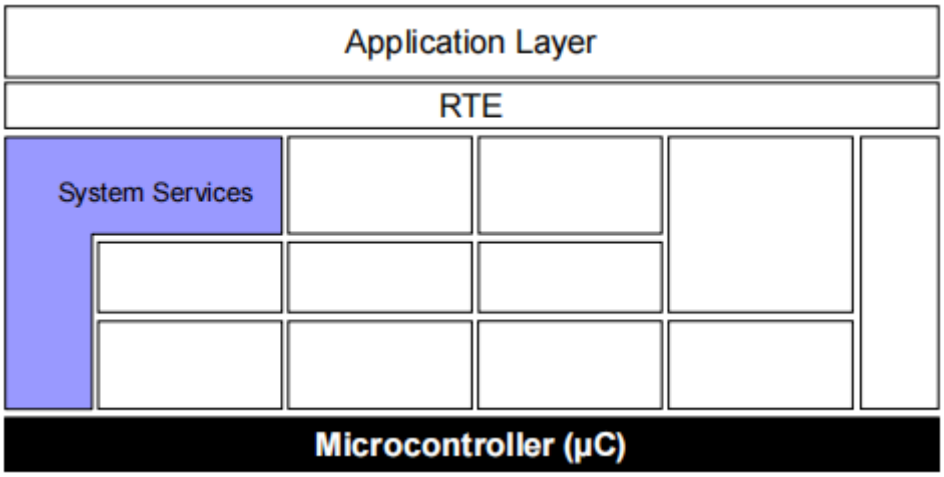
System Services是一组可以被所有层的模块使用的函数和模块。比如：实时操作系统（OS包含timer services）、Error Manager等待。

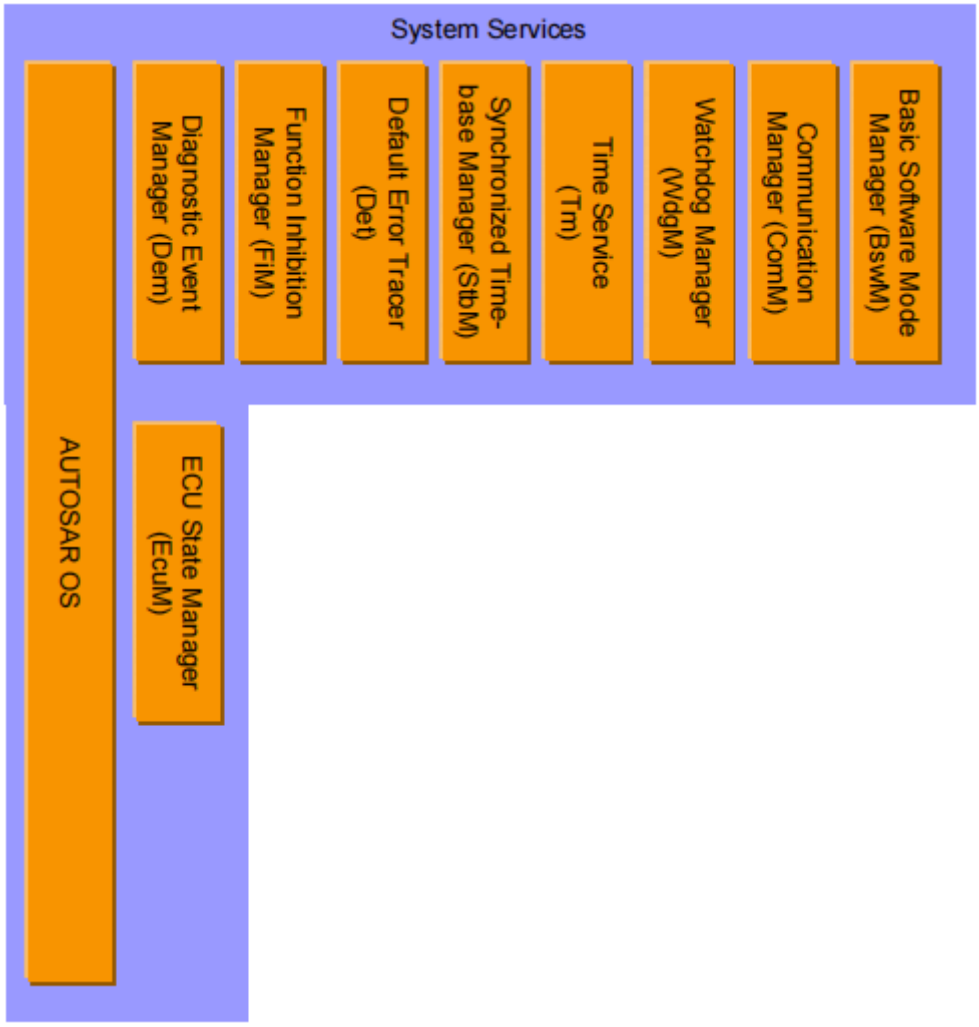
一些系统服务依赖于MCU（比如：OS），且可能支持特殊的MCU功能（比如：时间服务），此外也部分依赖于ECU硬件和应用软件。但是，有些系统服务也会独立于硬件和MCU。

主要作用：为应用软件和BSW提供基础服务。

实现依赖：部分依赖于MCU、ECU硬件和应用软件。

其上层接口独立于MCU和ECU硬件。

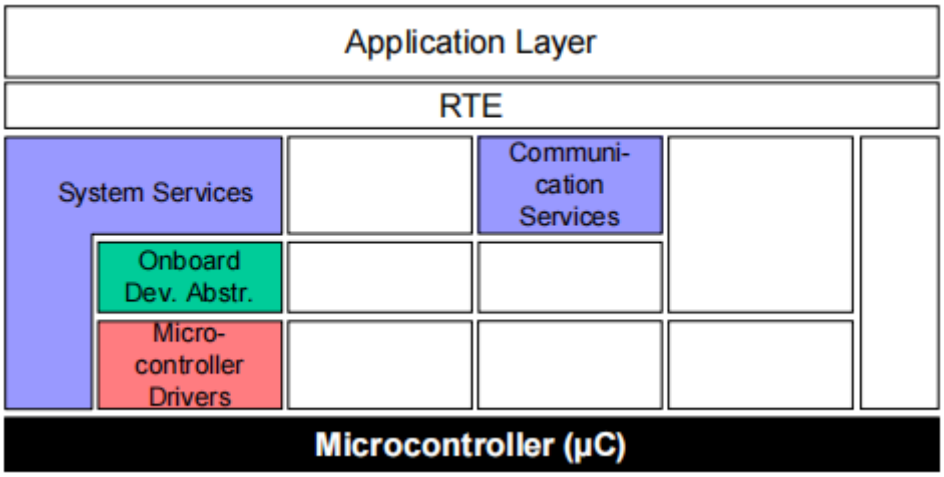


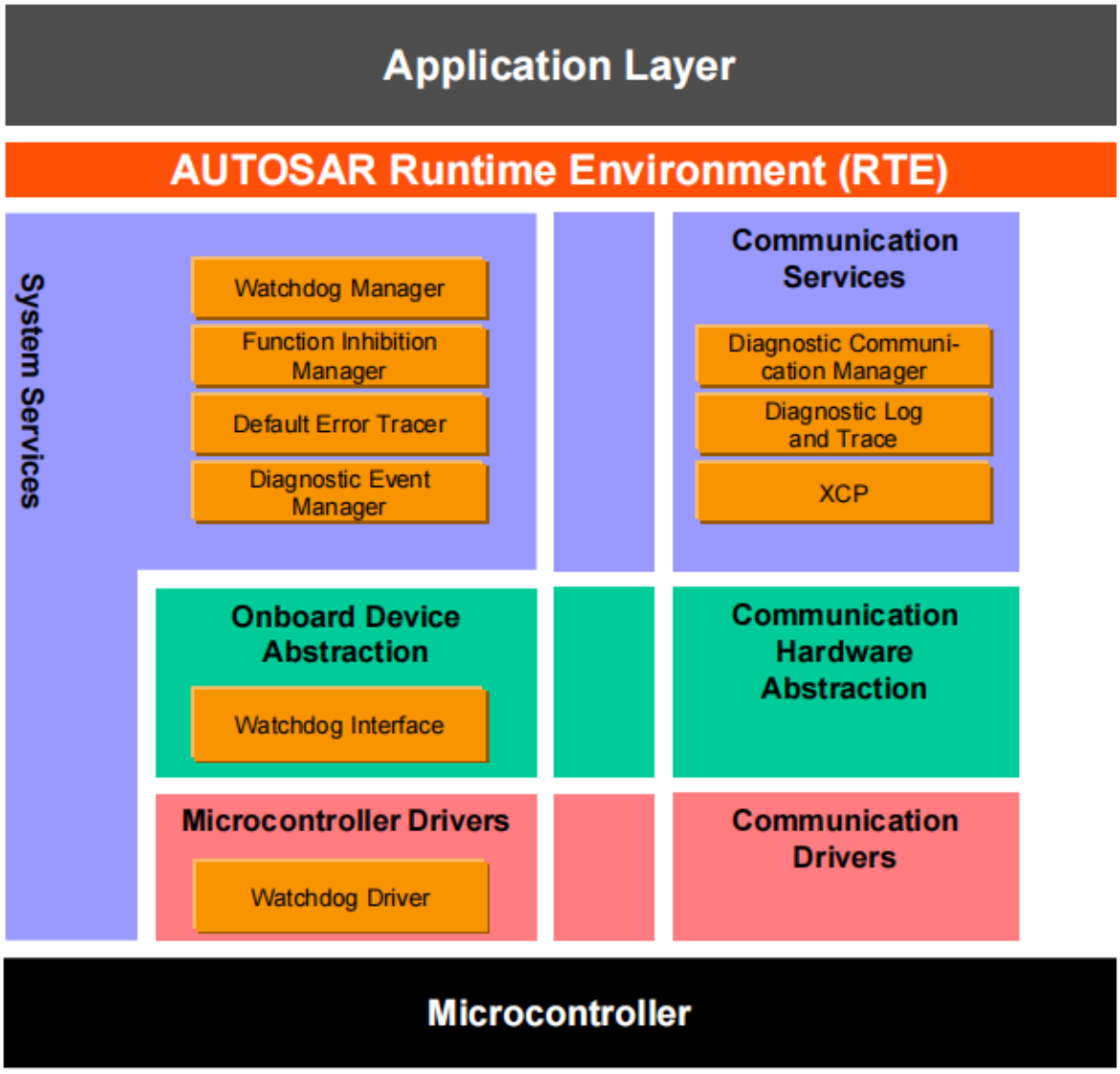


Error Handling, Reporting and Diagnostic

在AUTOSAR中有用于错误处理的不同方面的专用模块。比如：

- Diagnostic Event Manager（DEM）负责处理和保存诊断事件（错误）和相关的冻结帧数据。
- Diagnostic Log and Trace 模块支持应用软件的logging和tracing。它收集用户定义的日志信息，并将其转换为一种标准化的格式。
- 在基础软件中检测到的所有开发错误都会报告给Default Error Trace（DET）。
- Diagnostic Communication Manager 为诊断服务提供了一个通用的API。
- etc.





Application Layer: Sensor/Actuator Software Component

Sensor/Actuator AUTOSAR Software Component是一个用于传感器检测和执行器控制的特定类型的AUTOSAR软件组件。尽管它不属于AUTOSAR BSW，但其和本地信号的关系紧密。出于集成原因（标准化接口实现和接口描述），已决定将Sensor/Actuator SW Components定位在RTE之上。由于它们与原始局部信号的强相互作用关系，重定位受到限制。

主要作用：抽象连接到ECU的硬件传感器和执行器的特定物理属性。

实现依赖：独立于MCU和ECU硬件，但依赖于传感器和执行器。