
启明信息技术股份有限公司
QIMING INFORMATION TECHNOLOGY CO.,LTD
AUTOSAR分层架构介绍及
SmartSAR studio使用总结
打造汽车业IT第一品牌

- 内容介绍
- 一、什么是AUTOSAR
- 二、AUTOSAR分层概述
- 三、应用层
- 四、RTE层
- 五、基础软件层（BSW）
- 六、SmartSAR studio使用总结

一、什么是AUTOSAR

AUTOSAR是Automotive Open System Architecture（汽车开放系统架构）的首字母缩写，是一家致力于制定汽车电子软件标准的联盟。AUTOSAR是由全球汽车制造商、部件供应商及其他电子、半导体和软件系统公司联合建立，各成员保持开发合作伙伴关系。自2003年起，各伙伴公司携手合作，致力于为汽车工业开发一个开放的、标准化的软件架构。AUTOSAR这个架构有利于车辆电子系统软件的交换与更新，并为高效管理愈来愈复杂的车辆电子、软件系统提供了一个基础。此外，AUTOSAR在确保产品及服务质量的同时，提高了成本效率。

国内AUTOSAR研究情况

浙江大学ESE实验中心从2004年开始关注AUTOSAR，并率先加入了AUTOSAR组织。目前浙江大学ESE实验中心已经成功开发出一套符合AUTOSAR标准的集成的ECU开发工具链（简称为SmartSAR Studio），它可以用于ECU软件架构、网络系统配置、基础软件核配置、诊断、标定和仿真测试，支持从上到下、软件为中心的快速迭代开发模式。另外，ESE实验室中心已经开发出符合AUTOSAR标准的操作系统、通信等基础软件模块。

国外AUTOSAR研究情况

MICROSAR是Vector根据AUTOSAR标准开发的一系列产品级软件模块，包括RTE、CAL、OS、COM、IO、SYS和DIAG等等。在MICROSAR的帮助下，开发人员可以完全忽略硬件平台不同所带来的差异，甚至可以在缺少硬件平台的情况下先期开发应用程序，利用CANoe作为平台进行仿真和调试。这一切都是由于MICROSAR所提供的标准化接口。

MICROSAR需要DaVinci系列工具来进行配置。

DaVinci Developer：

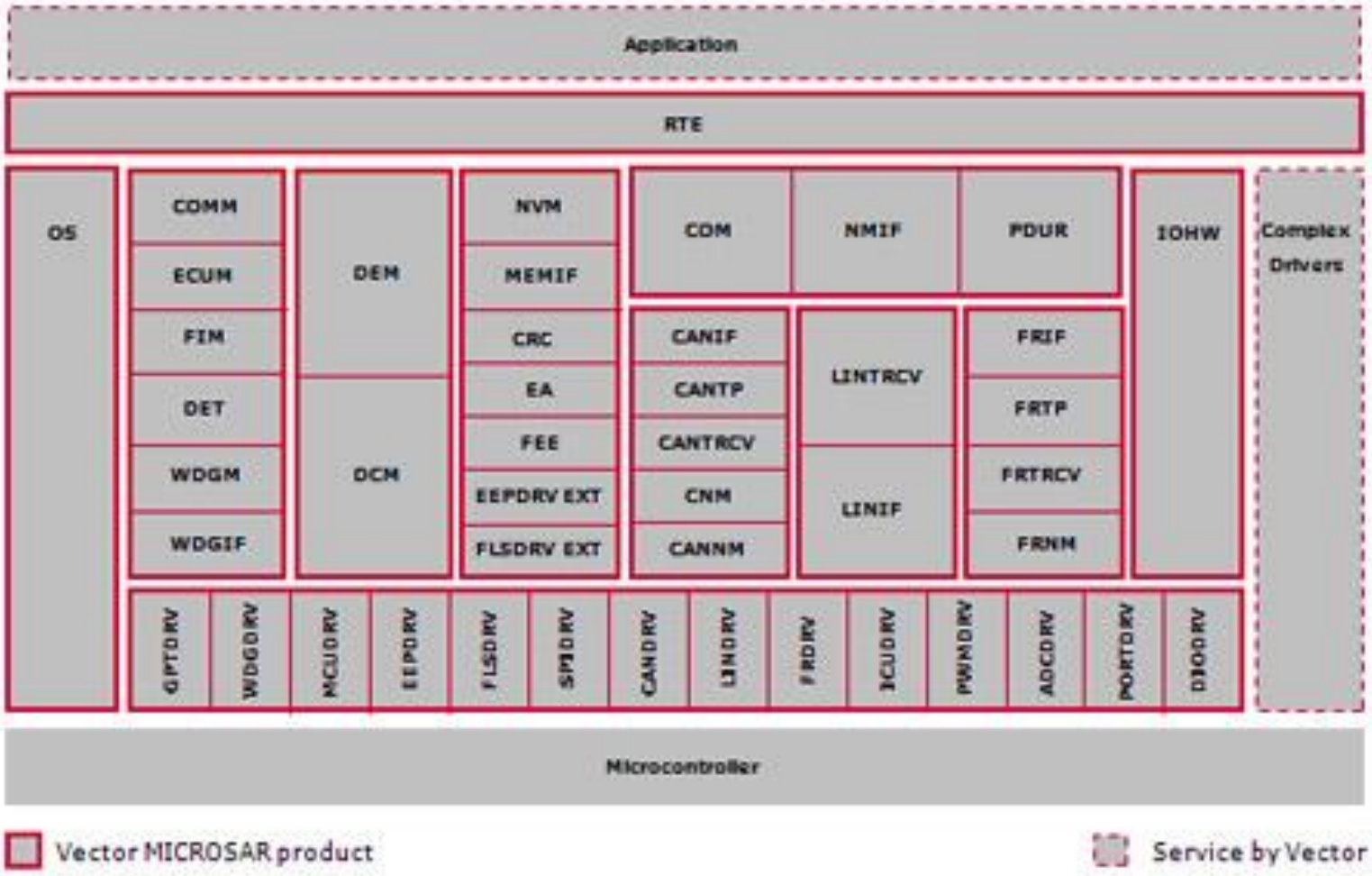
DaVinci Developer一个专用于符合AUTOSAR标准的ECU软件开发工具，它可以用来配置并生成ECU的RTE(RunTime Environment)源代码。用户可以利用DaVinci Developer的图形用户界面开发应用程序(SWC)以及定义应用程序接口，并可以方便的调用DaVinci系列的其他工具。

- 导入AUTOSAR的ECU交换文档(Extract of ECU Description File)
- 图形化定义软件组件(SWC)
- 定义端口(Ports)和数据类型(Data Elements)
- 将运行实体(Runnables)映射到操作系统任务(Task)中
- 导入/导出AUTOSAR的arxml文件
- 从网络数据库中导入信号
- 针对ECU配置的一致性校验
- 与Matlab/Simulink无缝集成

DaVinci Configurator Pro:

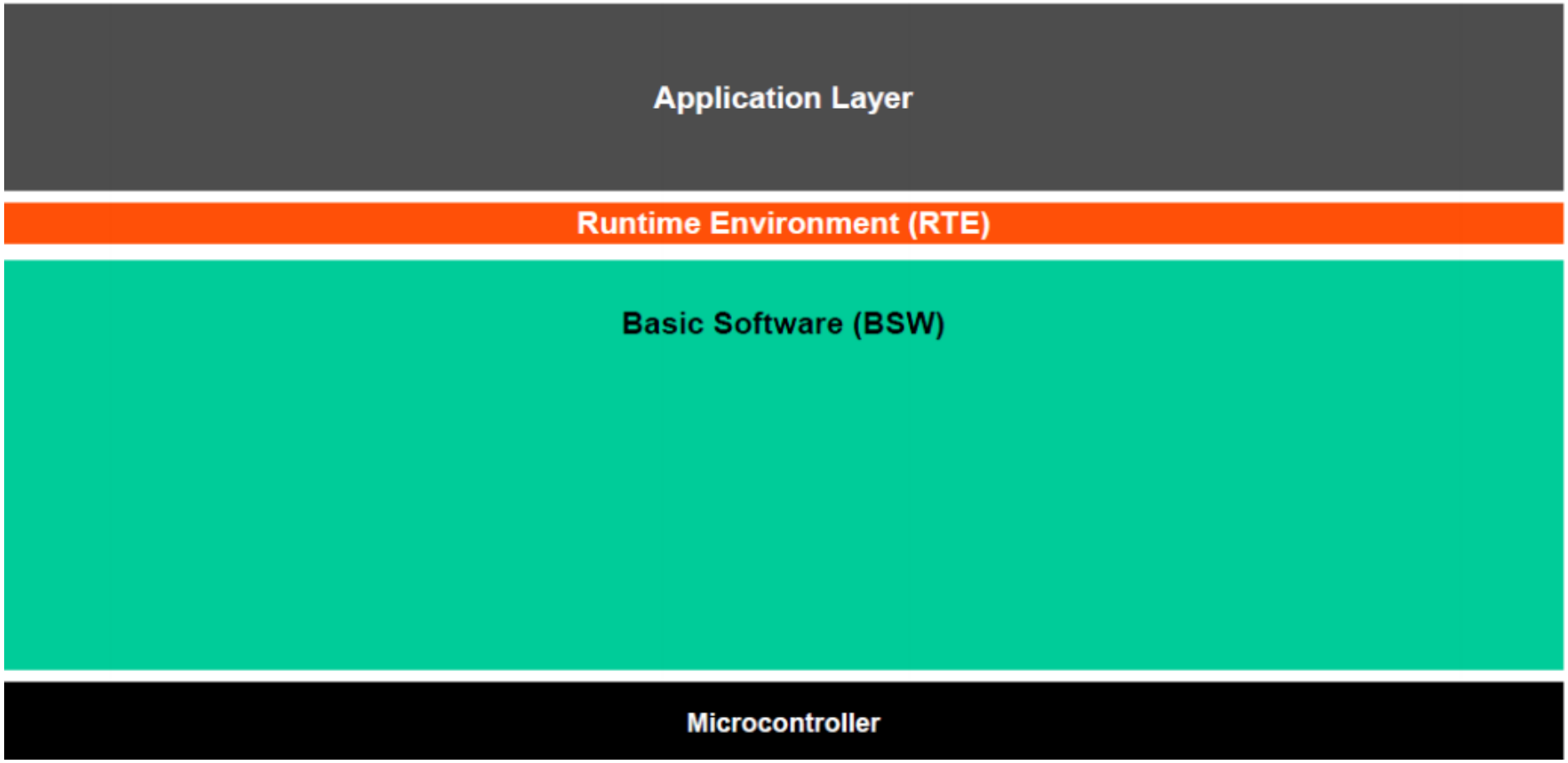
DaVinci Configurator Pro是一个符合AUTOSAR标准的软件配置工具，它专门用于配置并生成ECU中的Basic Software(BSW)。它能保证在配置各底层软件模块的过程中，各配置参数的一致性。如果出现配置数据错误或缺失，DaVinci Configurator Pro能及早发现并提出警告。

- 使用图形化的配置简化了各参数间复杂的内部关系
- 支持在同一系统中并行配置不同版本的BSW(如2.1和3.0)
- 基于AUTOSAR规范的验证过程
- 依然使用GENy来配置通信相关模块(为CANbedded用户带来方便)
- 针对BSW配置的一致性校验


Application
RTE
OS
COMM
ECUM
FIM
DET
WDGM
WDGIF
DEM
DCM
NVM
MEMIF
CRC
EA
FEE
EEPDRV EXT
FLSDRV EXT
COM
NMIF
PDUR
CANIF
CANTP
CANTRCV
CNM
CANNM
LINTRCV
LINIF
FRIF
FRTP
FRTRCV
FRNM
IOHW
Complex Drivers
GPTDRV
WDGDRV
MCUDRV
EEPDRV
FLSDRV
SPIDRV
CANDRV
LINDRV
FRDRV
ICUDRV
PWMDRV
ADCDRV
PORTDRV
DIODRV
Microcontroller
Vector MICROSAR product
Service by Vector

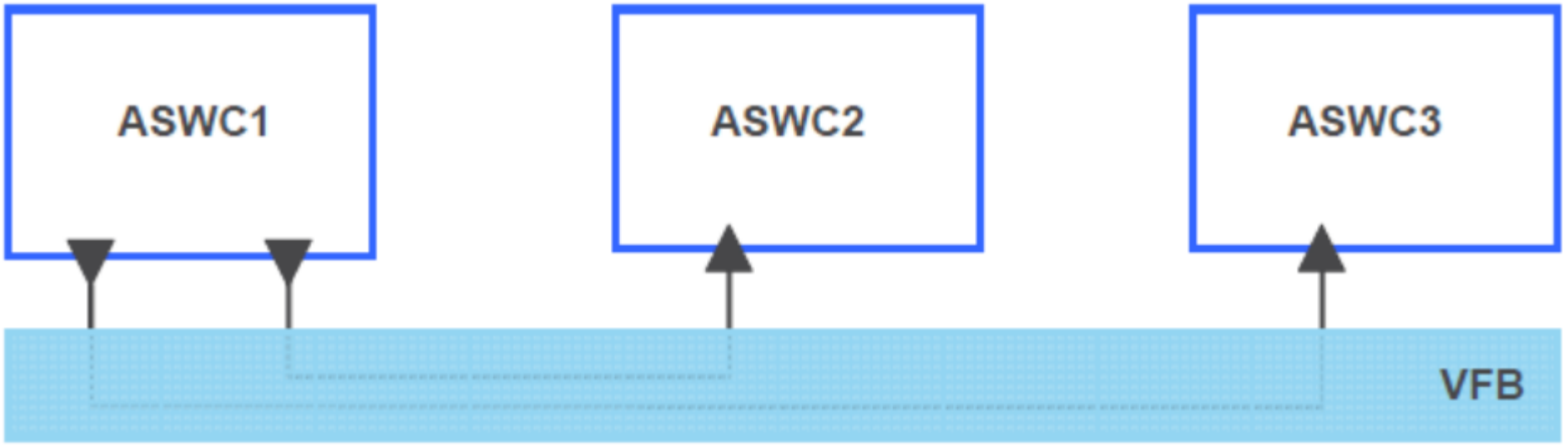
二、AUTOSAR的分层概述

将运行在Microcontroller之上的ECU软件分为Application、 RTE、 BSW三层



- **1、应用层**

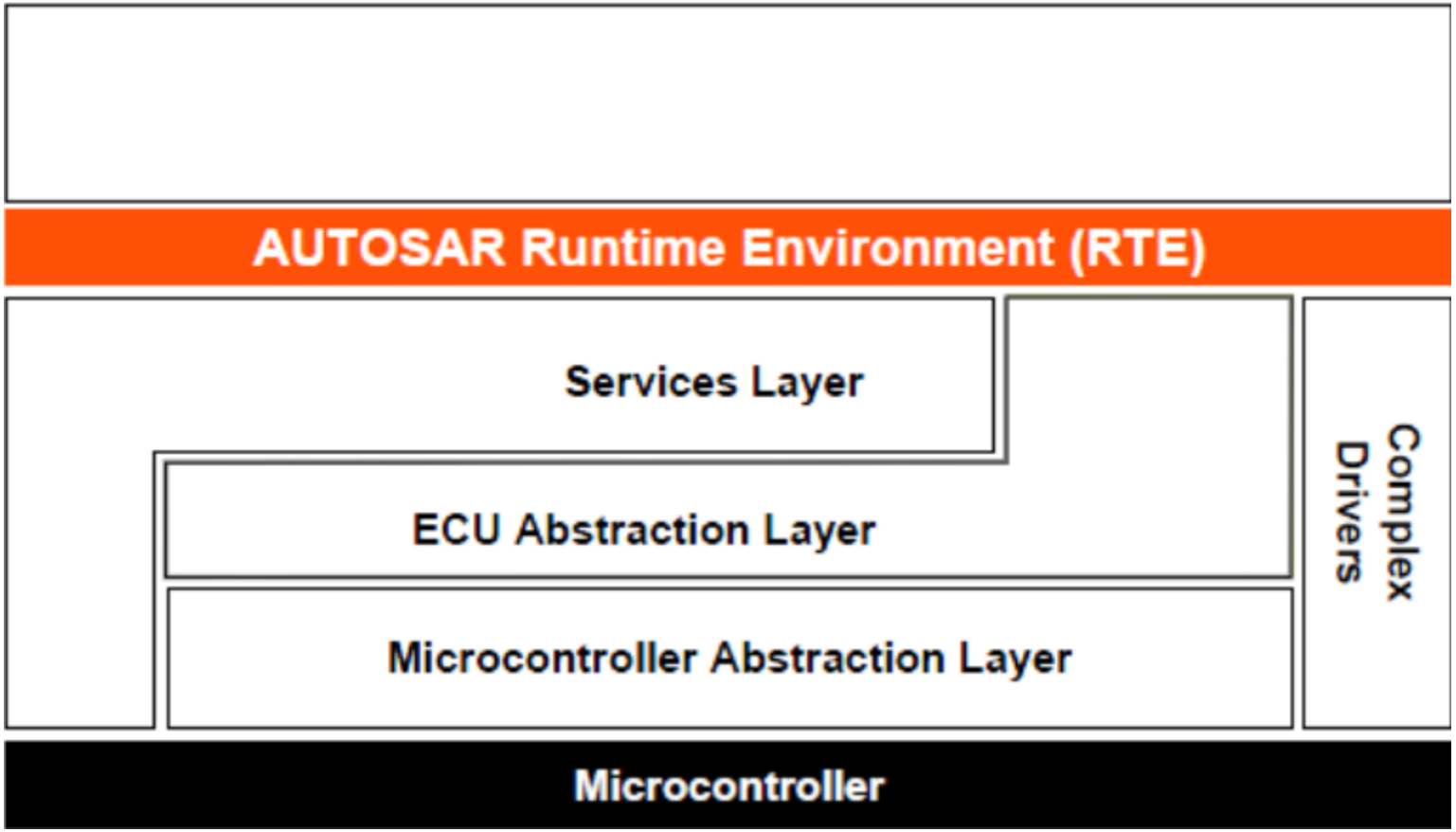
  应用层将软件都划分为一个Atomic Software component（ASWC），包括硬件无关的Application Software Component、 Sensor Software Component、 Actuator Software Component等。



2、RTE层

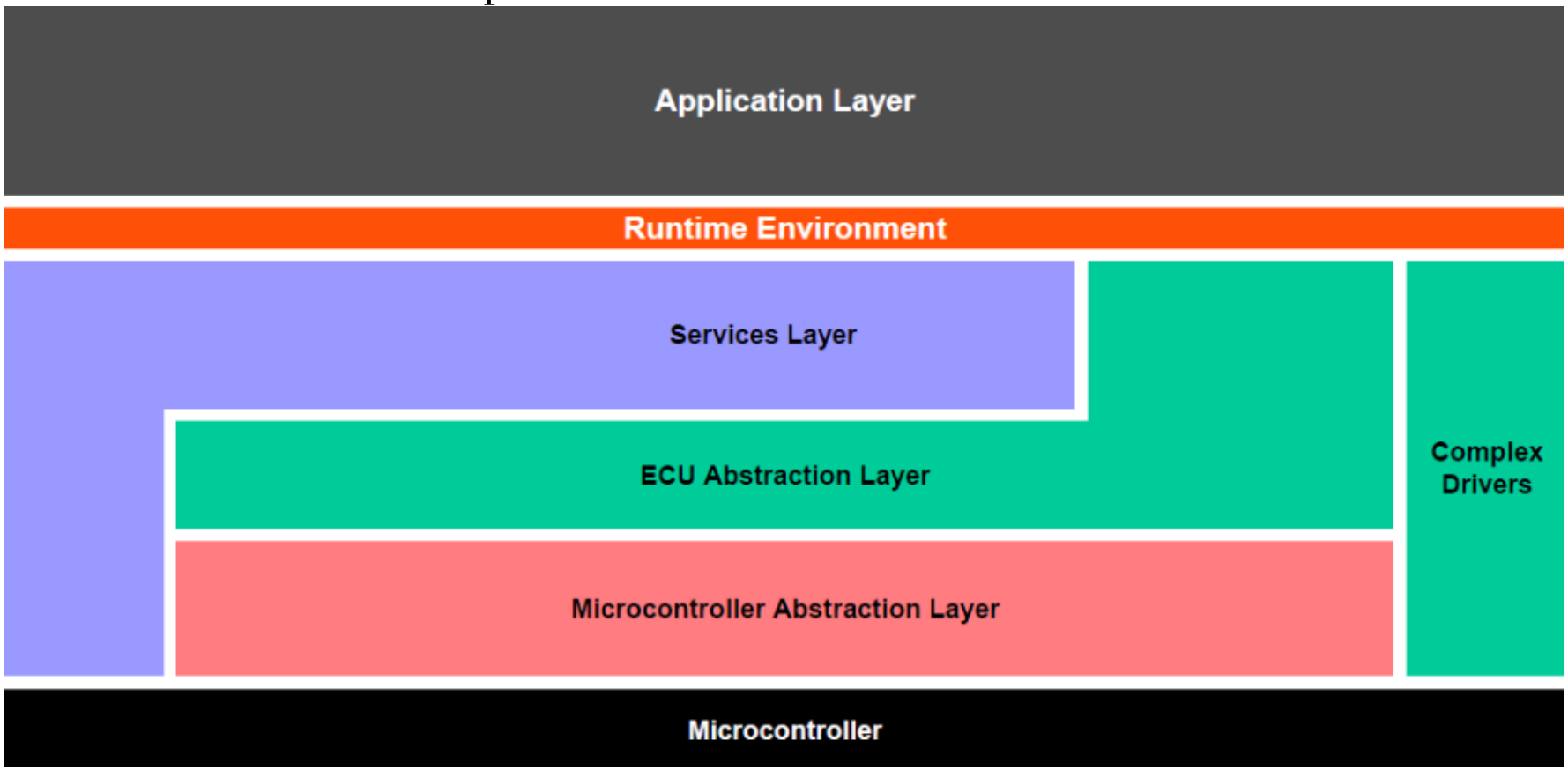
RTE提供基础的通信服务，支持Software Component之间和Software Component到BSW的通信（包括ECU内部的程序调用、ECU外部的总线通信等情况）。

RTE使应用层的软件架构完全脱离于具体的单个ECU和BSW。

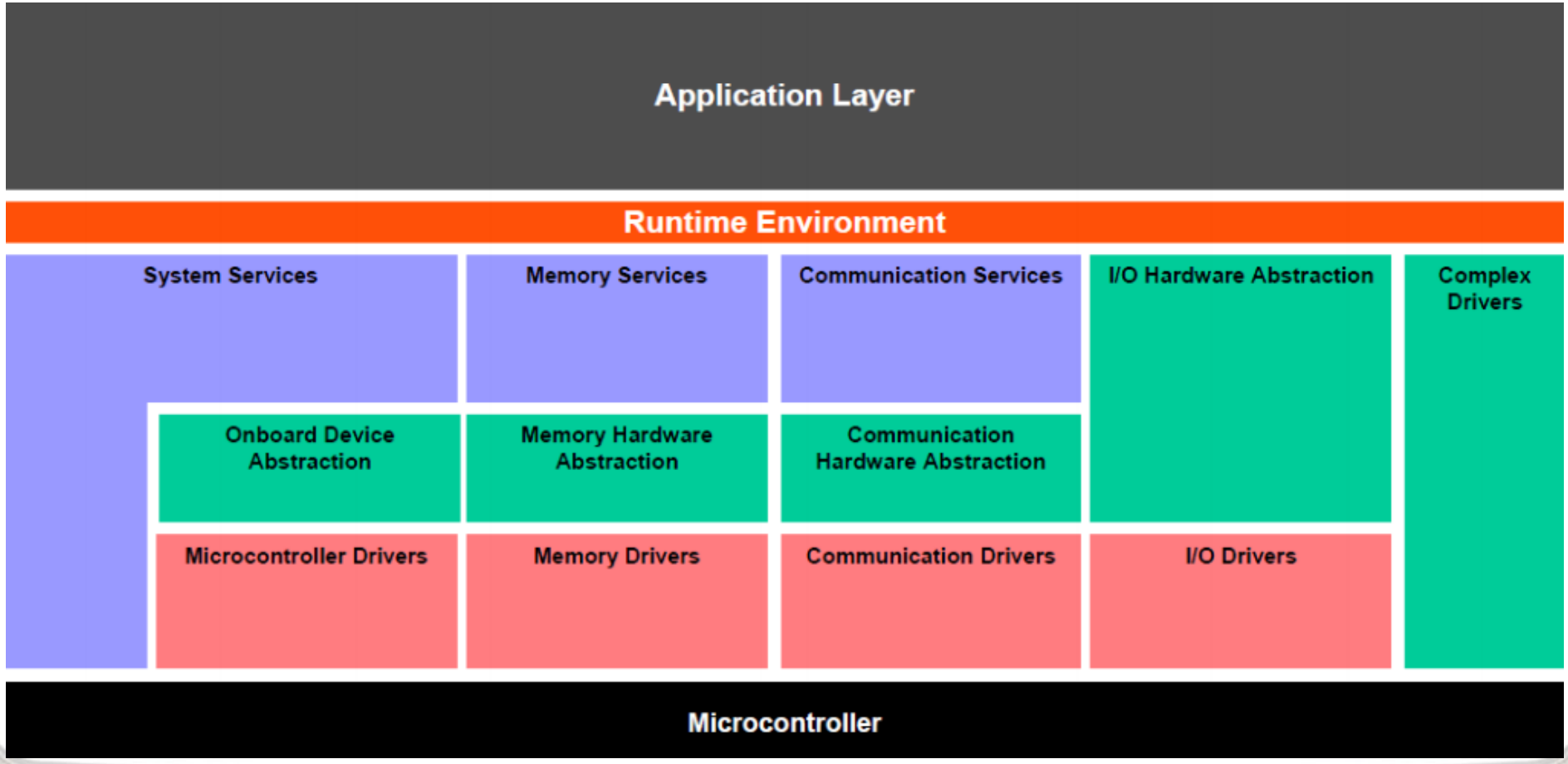

AUTOSAR Runtime Environment (RTE)
Services Layer
ECU Abstraction Layer
Microcontroller Abstraction Layer
Complex Drivers
Microcontroller

- # 3、BSW层

将基础软件层(BSW)分为Service、ECU Abstraction、Microcontroller Abstraction以及Complex Drivers。

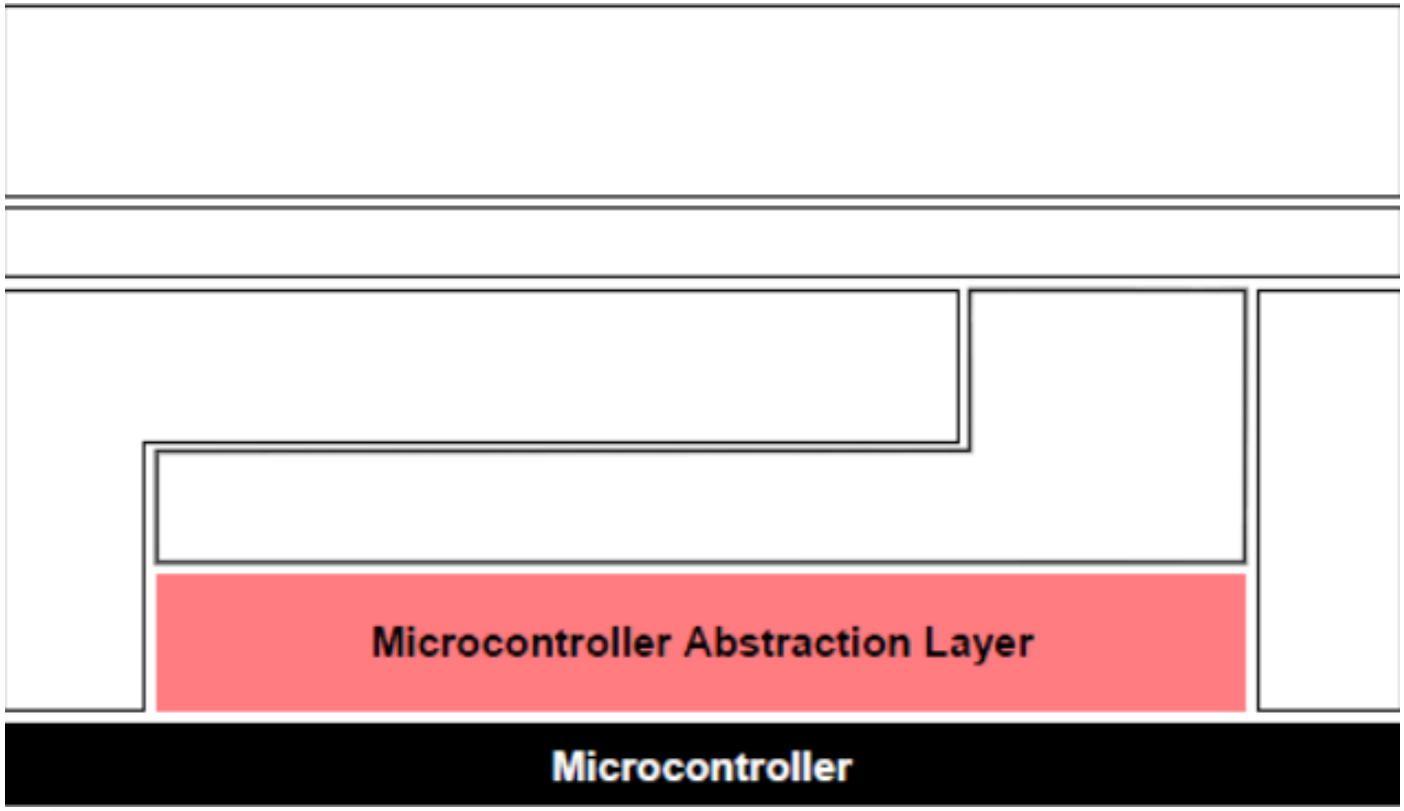


每层的BSW中包括不同的功能模块。比如Service层包括系统服务、内存服务、通信服务。



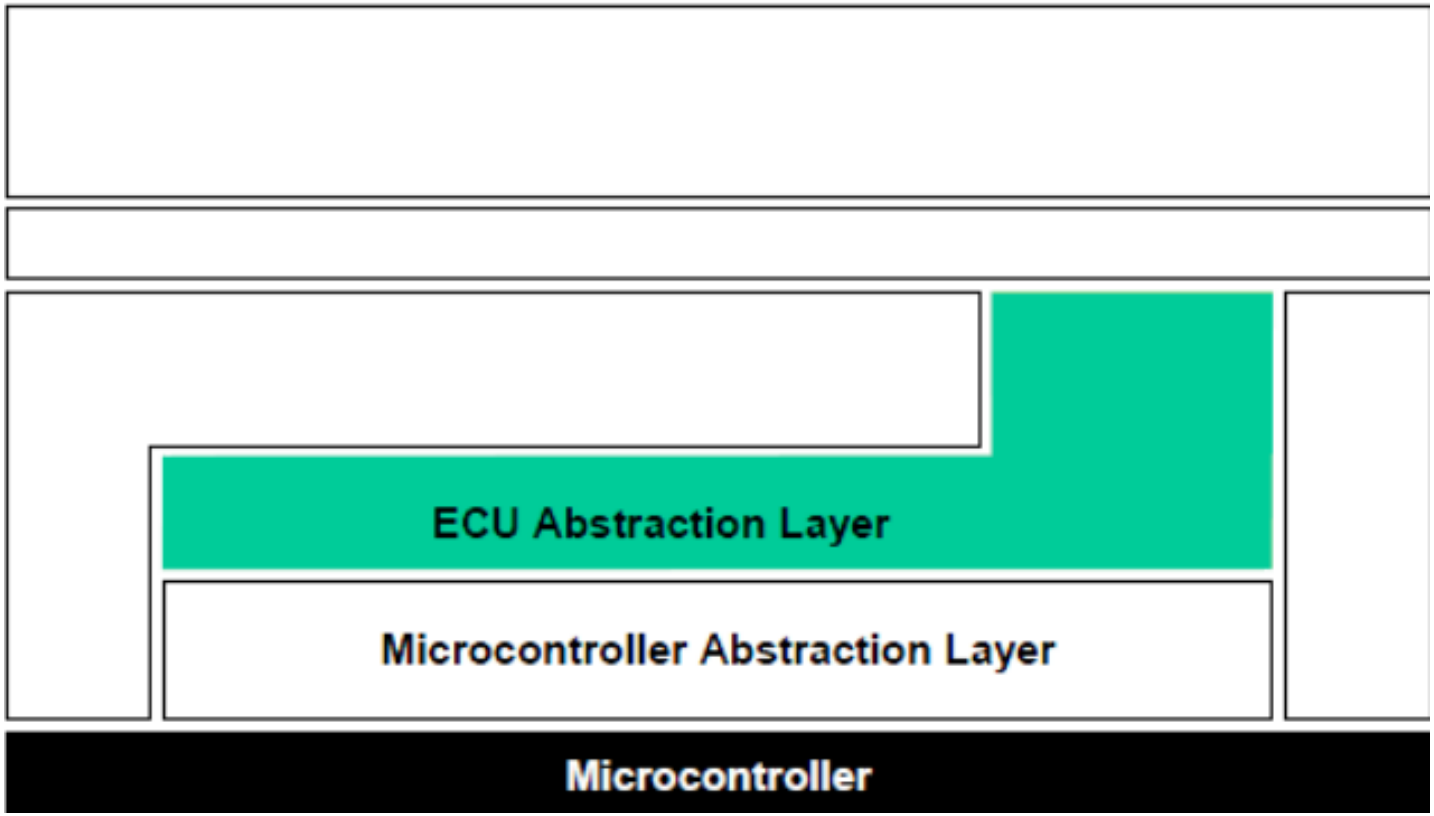
微控制器抽象层(Microcontroller AbstractionLayer)是在BSW的最底层，它包含了访问微控制器的驱动。

微控制器抽象层使上层软件与微控制器相分离，以便应用的移植。

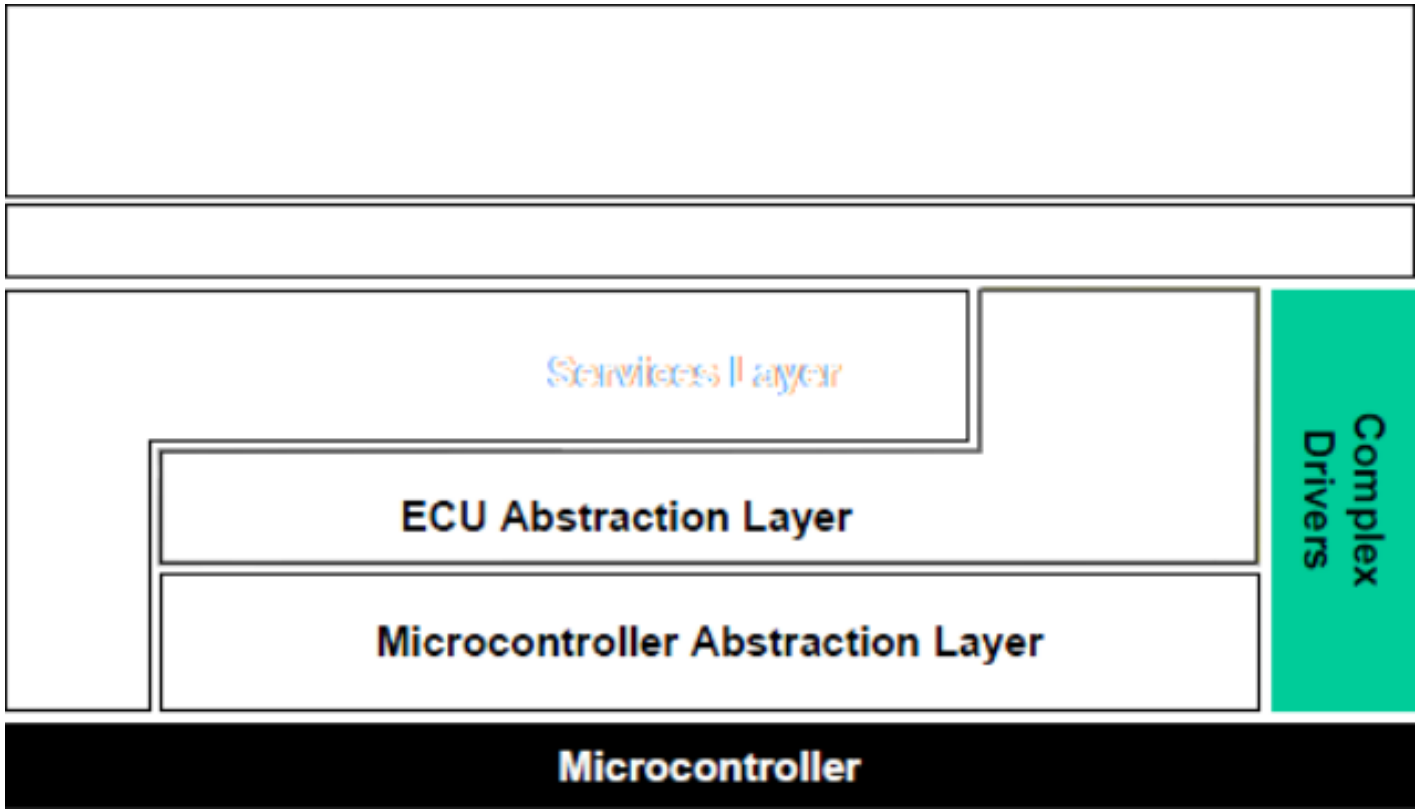


ECU抽象层封转了微控制器层以及外围设备的驱动。

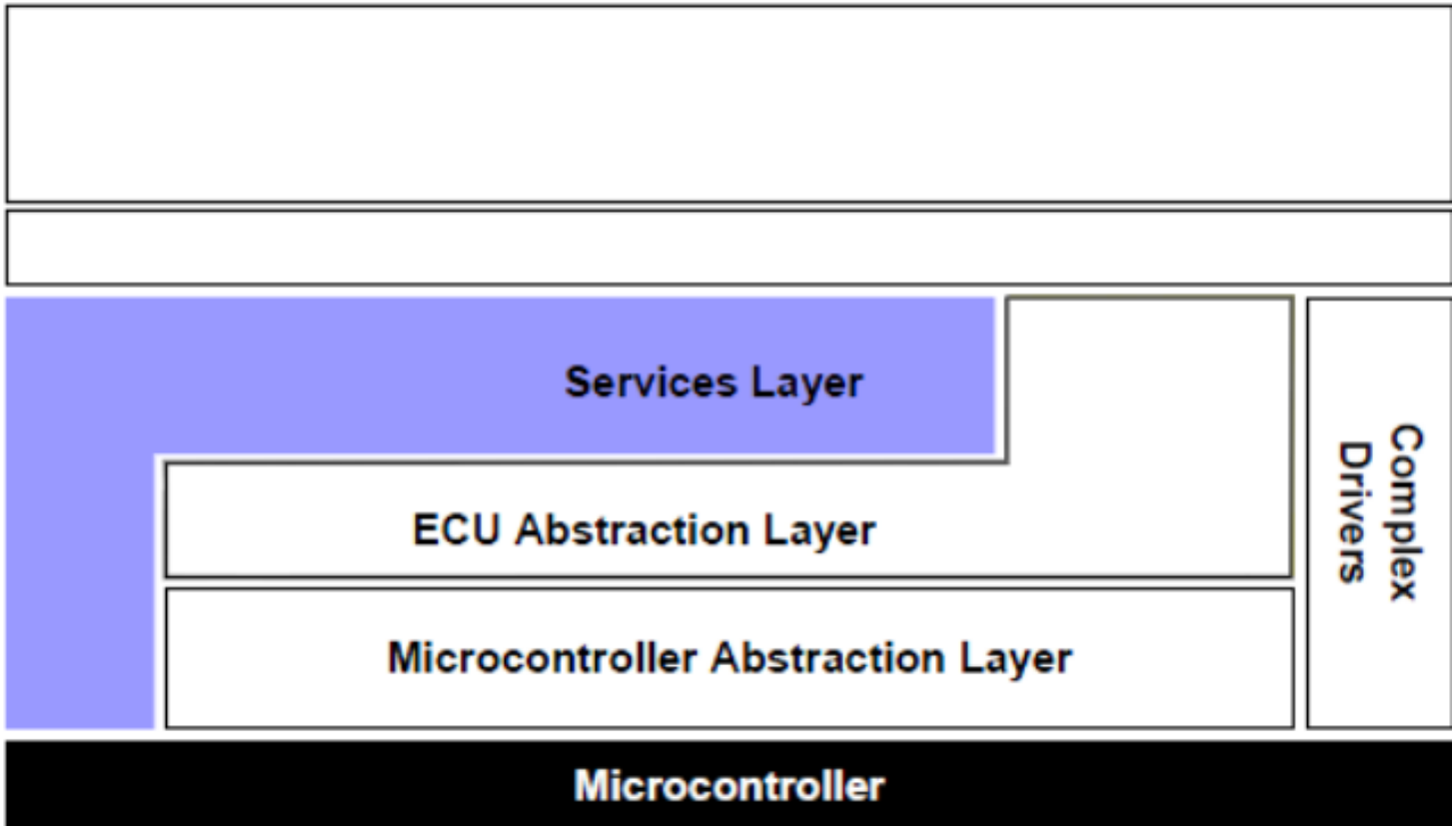
将微控制器内外设的访问进行了统一，使上层软件应用与ECU硬件相剥离。



为了满足实时性等要求，可以利用复杂驱动（Complex Drivers），让应用层通过RTE直接访问硬件。
也可以利用复杂驱动封转已有的非分层的软件，以实现向AUTOSAR软件架构逐步实施。



服务层（Service Layer）位于BSW的最上面，将各种基础软件功能以服务的形式封转起来，供应用层调用。
服务层包括了RTOS、通信与网络管理、内存管理、诊断服务、状态管理、程序监控等服务。



BSW包括以下服务类型：

Input/output(I/O)服务：将执行器、传感器以及外设的访问标准化

内存服务：将微控制器内外内存的访问进行统一封转

通信服务：将整车网络系统、ECU网络系统、软件组件内的访问进行统一封转

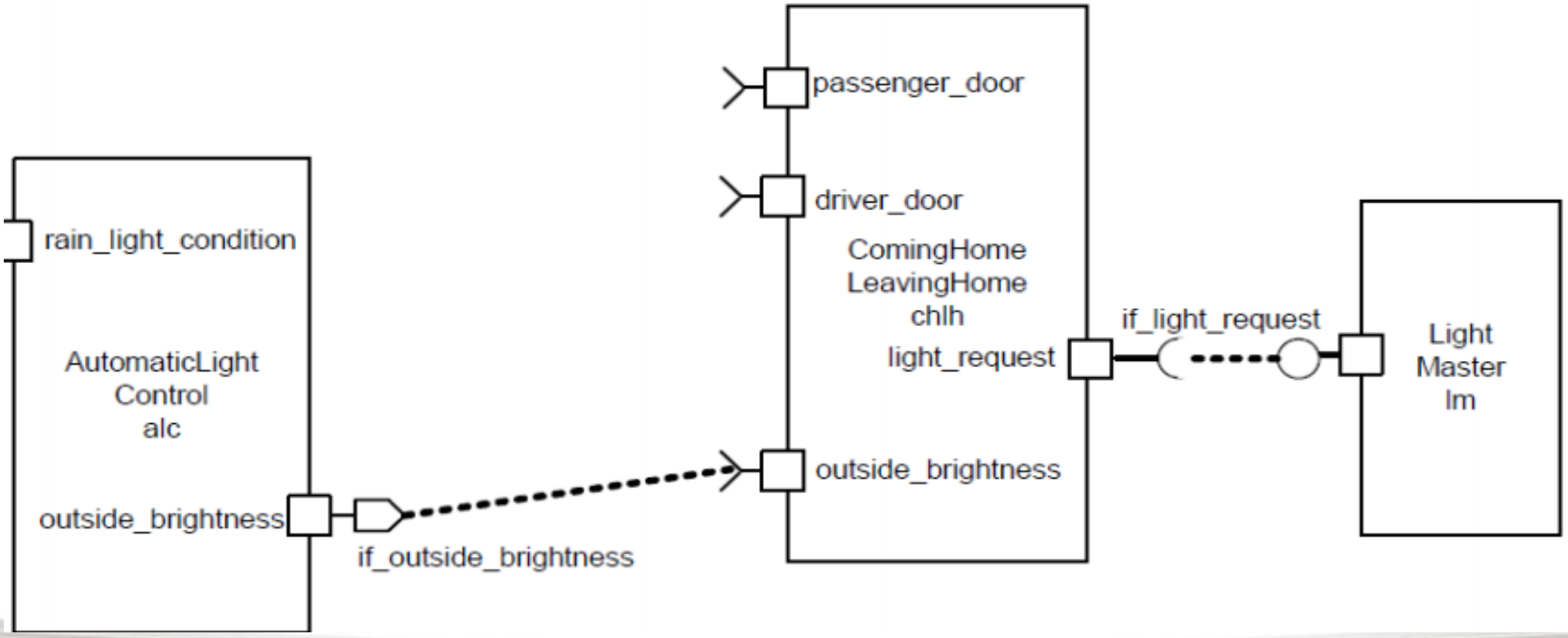
系统服务：包括RTOS、定时器、错误处理、看门狗、状态管理等服务

二、应用层

应用层由各种AUTOSAR Software Component（SW-C）组成

每个AUTOSAR　SW-C都封转了各种应用的功能集，可大可小

每个AUTOSAR SW-C只能运行在一个ECU中，也可称为AtomicSW-



可以通过算法建模、手写代码等多种方式实现SW-C。

在AUTOSAR架构体系中，SW-C的实现：

与MCU类型无关

与ECU类型无关

与相互关联的SW-C的具体位置无关

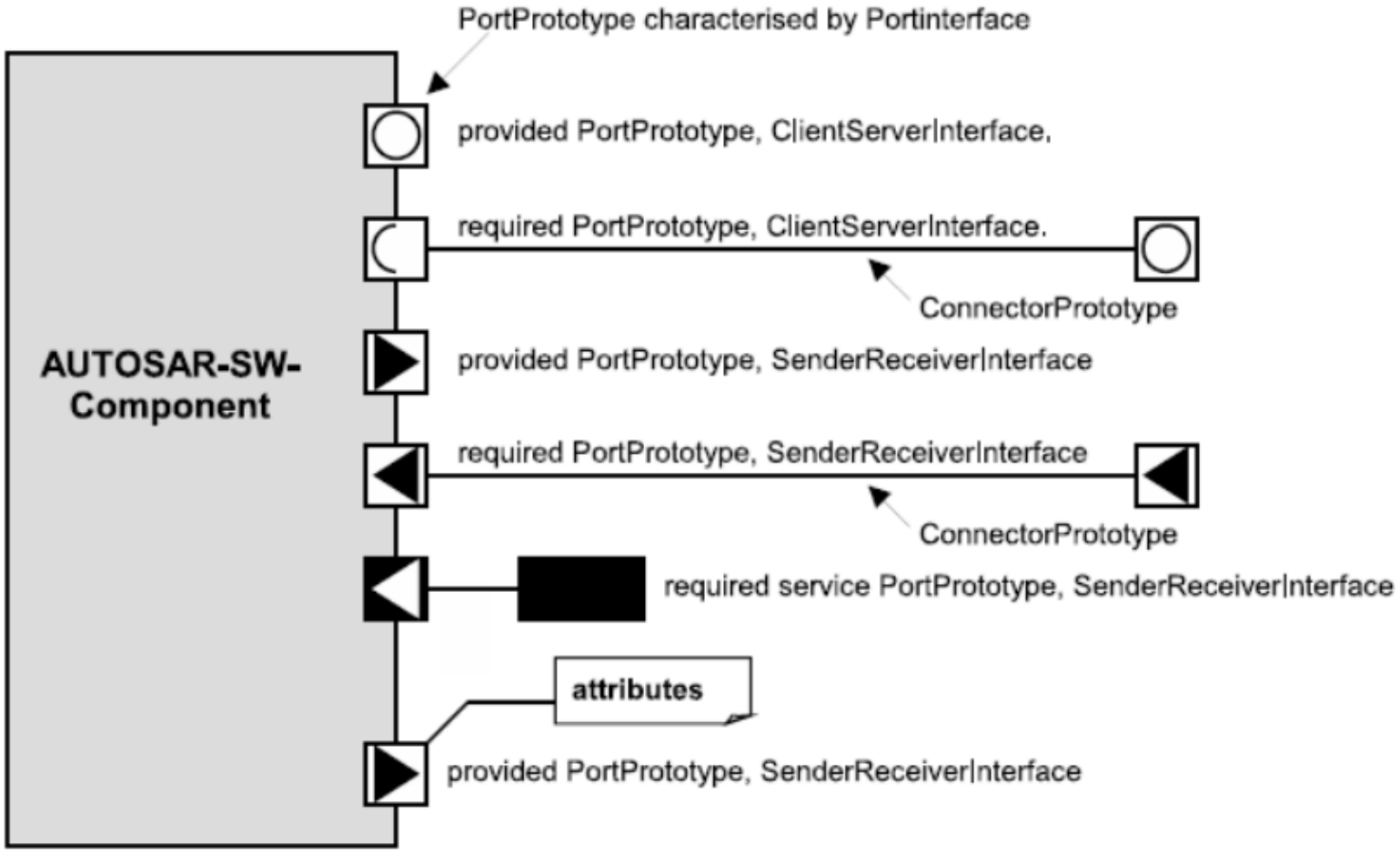
与具体SW-C的实例个数无关

Software Component Template规定了SW-C的描述规范

Port和Interface：

Port：表示输入（RPort）或输出（PPort）

Interface：具体输入输出的方式、数据类型等



SW-C的类型：

软件组件单元（ ASWC ）

应用软件组件

输入输出软件组件

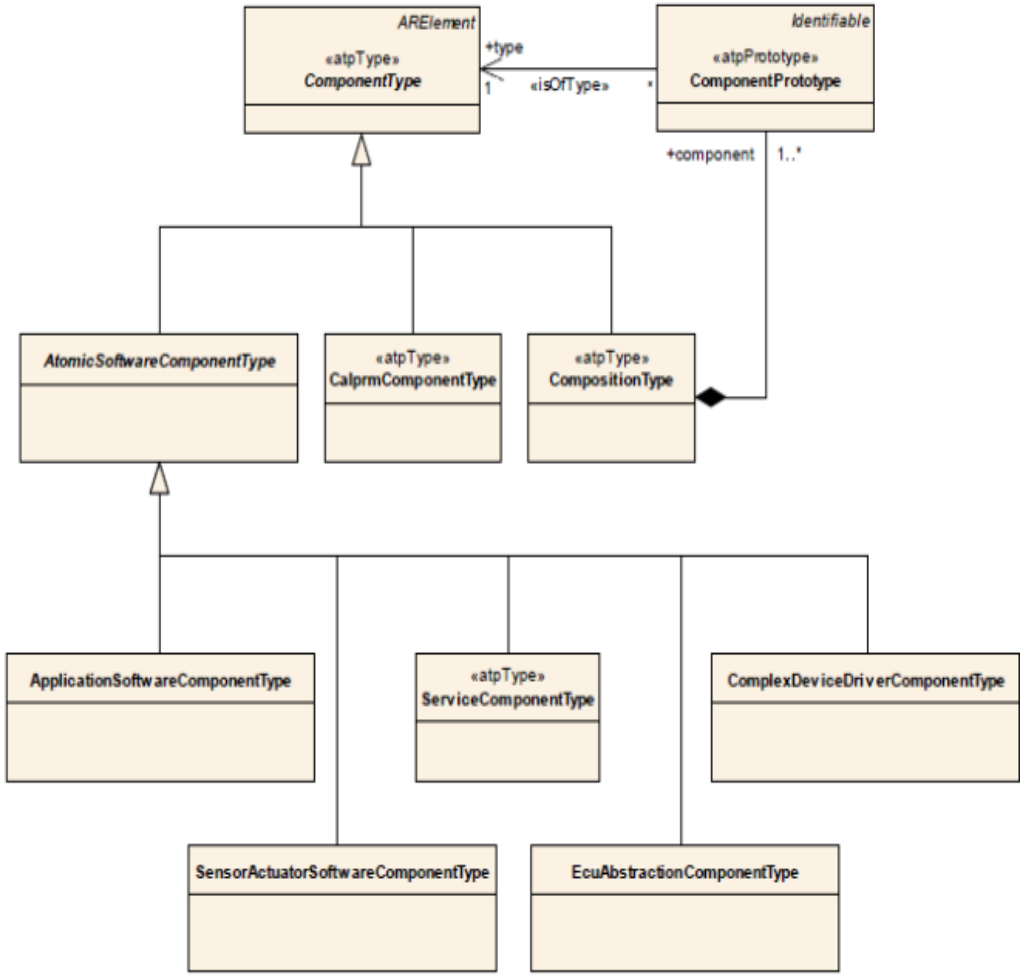
服务组件

ECU抽象组件

复杂驱动组件

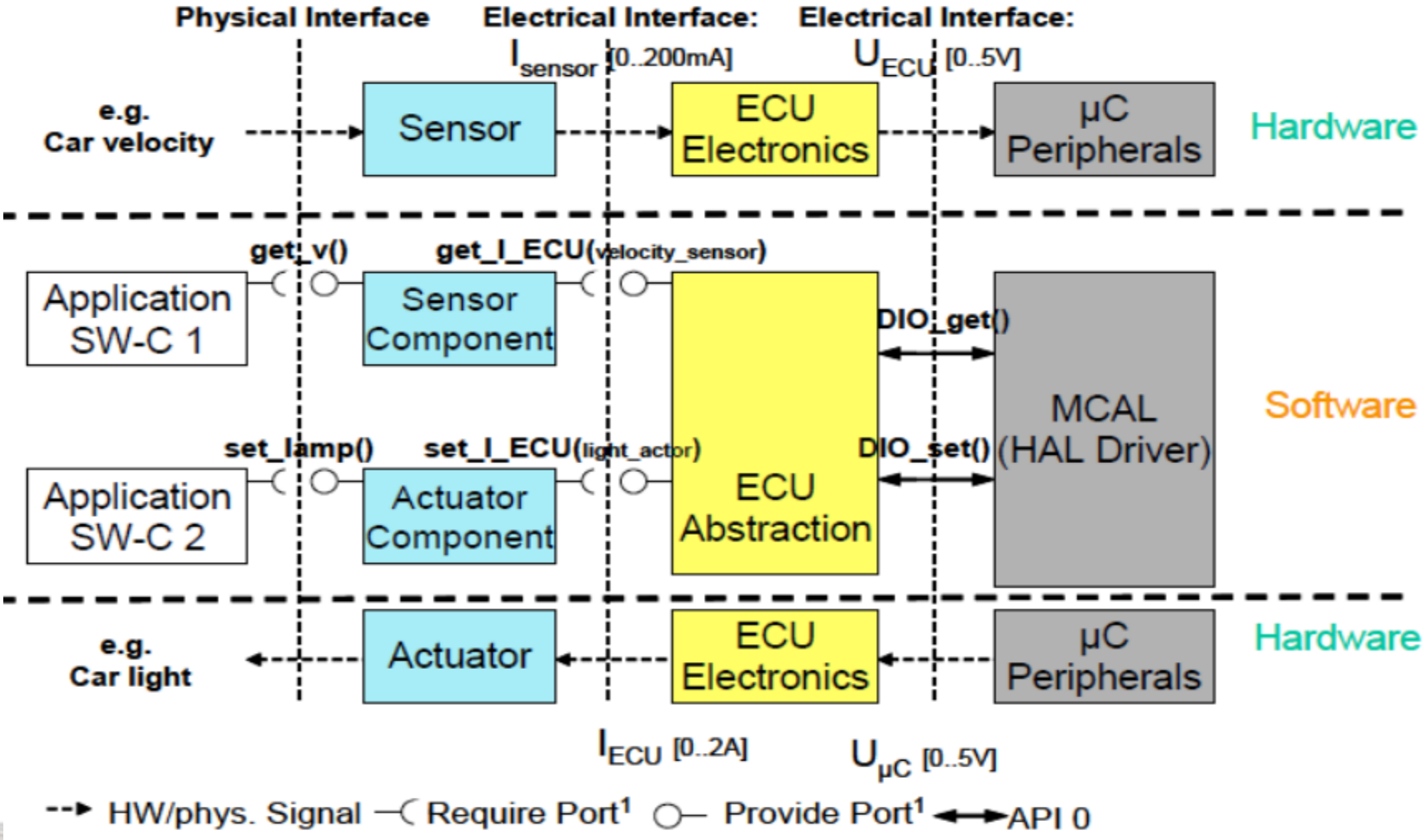
标定程序组件

组合Composition



Sensor/Actuator Software Components：

所有I/O的输入输出都通过Sensor/Actuator SW-C



Composition：

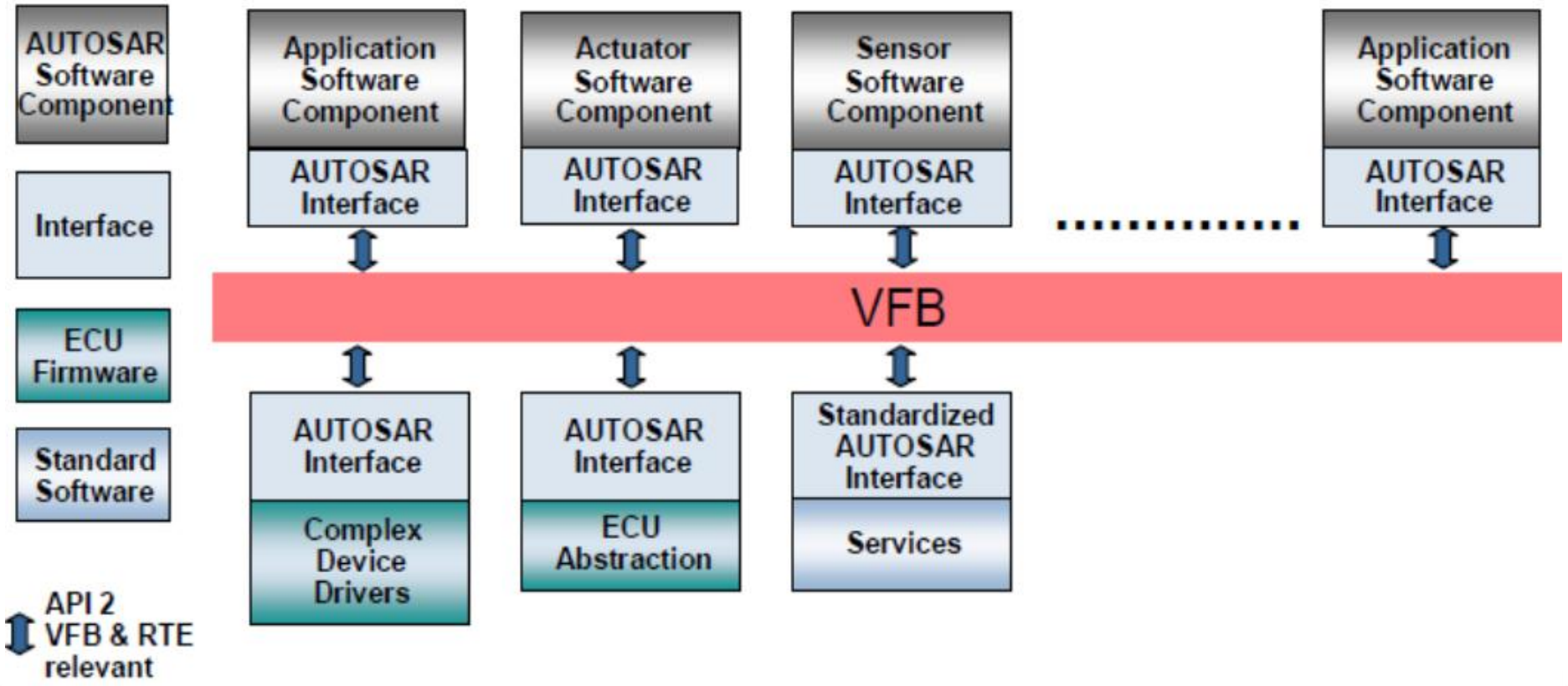
Composition是多个ASWC的实例集合，也当做是SW-C。

Composition的Port是内部某个ASWC的Port代理，通过DelegationConnector来表示。

Composition内ASWC之间的输入输出是通过AssemblyConnector来表示。

Virtual Functional Bus：

所有的Component（包括ASWC、ECU抽象、服务、复杂驱动）之间的通信组成了VFB。



Runtime Environment ：

RTE是VFB在具体一个ECU中的实例。

RTE实现了应用层SW-C之间、应用层SW-C与BSW之间的具体通信。

RTE通过划分RTOS的任务、资源、事件等，提供给组件一个隔离底层中断的运行时环境。

RTE的通信实现：

SW-C之间的通信是调用RTE API函数而非直接实现的，都在RTE的管理和控制之下。

每个API遵循统一的命名规则且只和软件组件自身的描述有关。

具体通信实现取决于系统设计和配置，都由工具供应商提供的RTE Generator自动生成的。

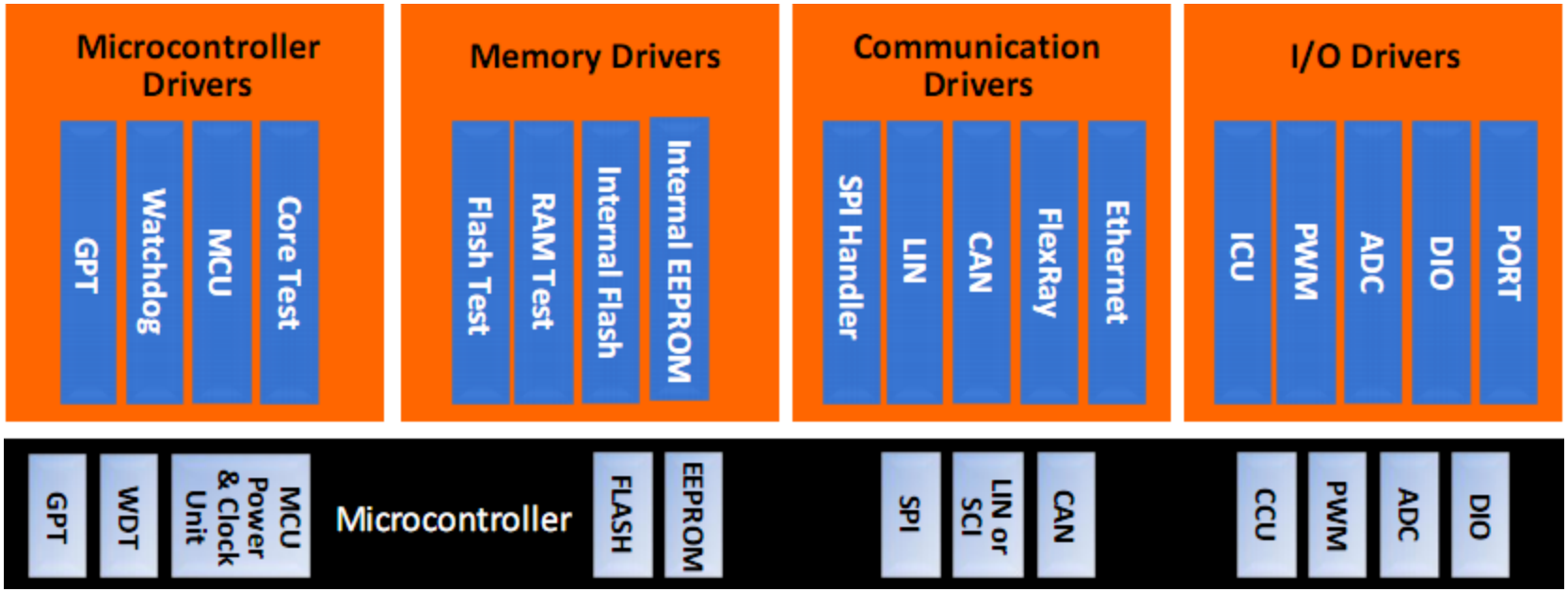
微控制器抽象层：

通信驱动：SPI、CAN等。

I/O驱动：ADC、PWM、DIO等。

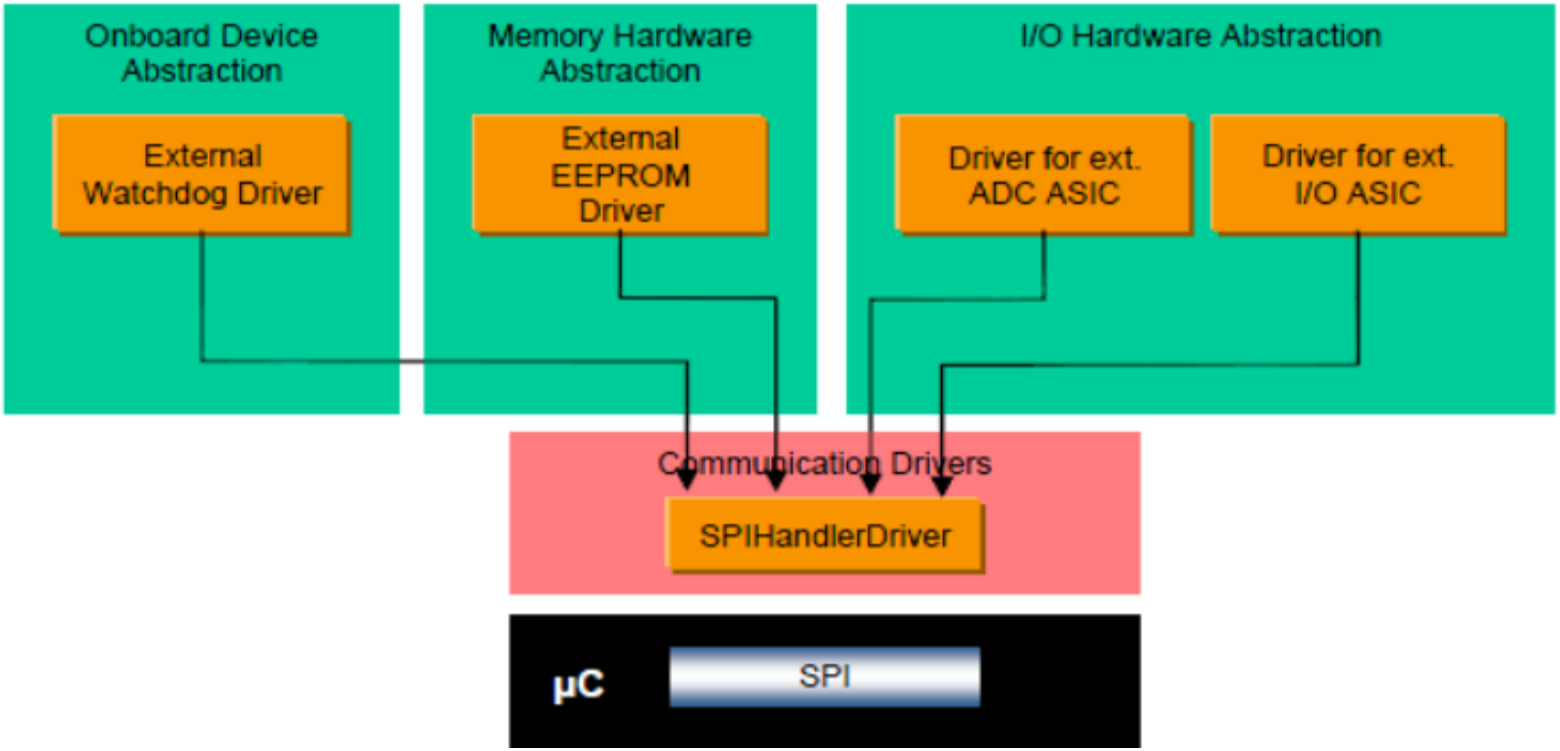
内存驱动：片内EEPROM、Flash等。

微控制器驱动：Watchdog、GPT等。



微控制器抽象层：SPIHandlerDriver

SPIHandlerDriver封转了统一访问SPI总线的接口，上层软件可以并发的多个访问。



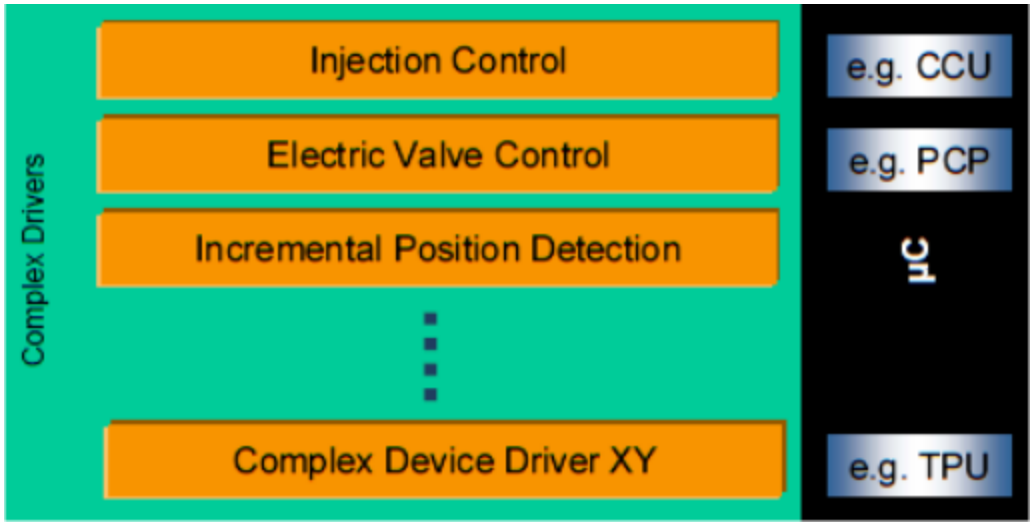
复杂驱动：

利用中断、TPU、PCP等，复杂驱动可以实现实时性高的传感器采样、执行器控制等功能。

比如：注射控制；

电子阀门控制；

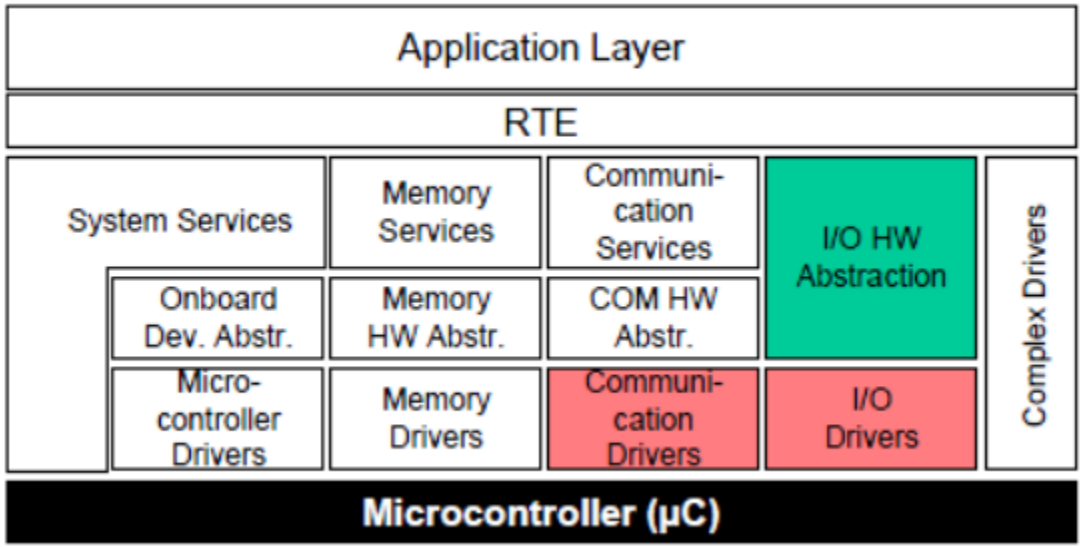
增量位置测量监测；

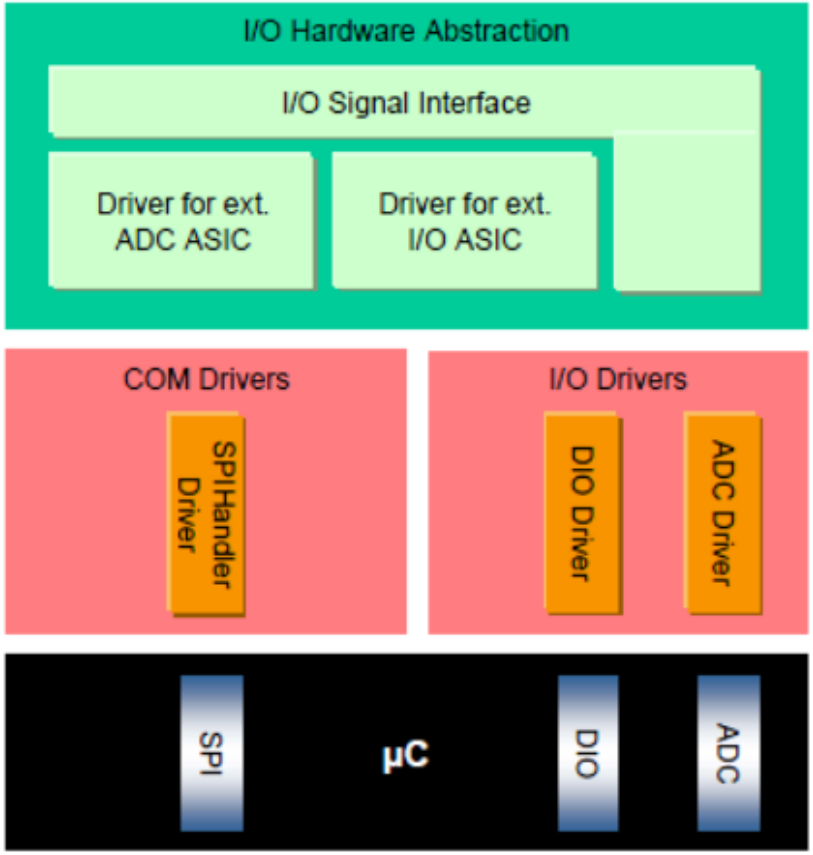


I/O硬件抽象：

可以通过I/O硬件抽象中的信号接口来访问不同的I/O设备。

将I/O信号都进行了封转，比如电流、电压等。

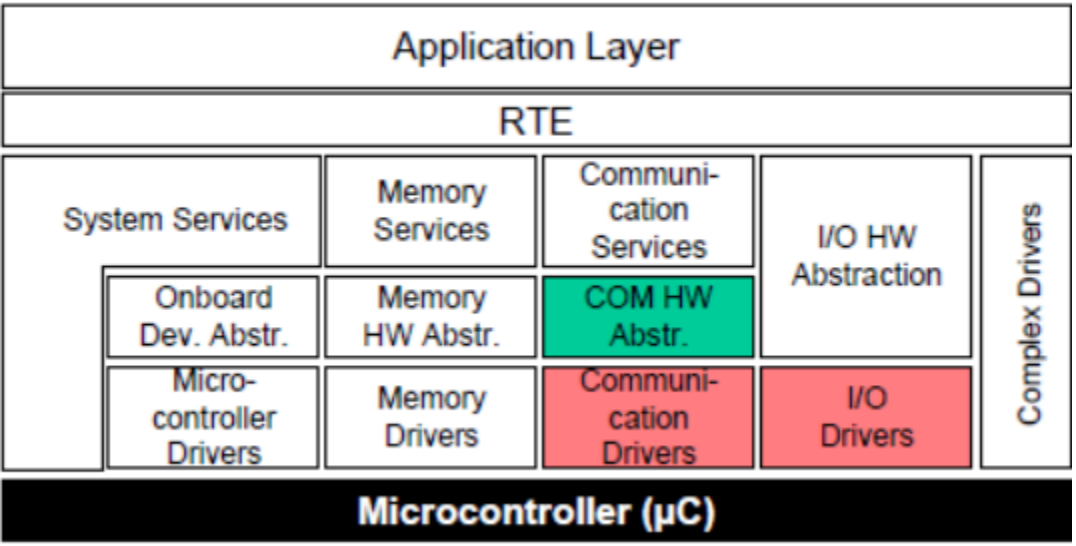


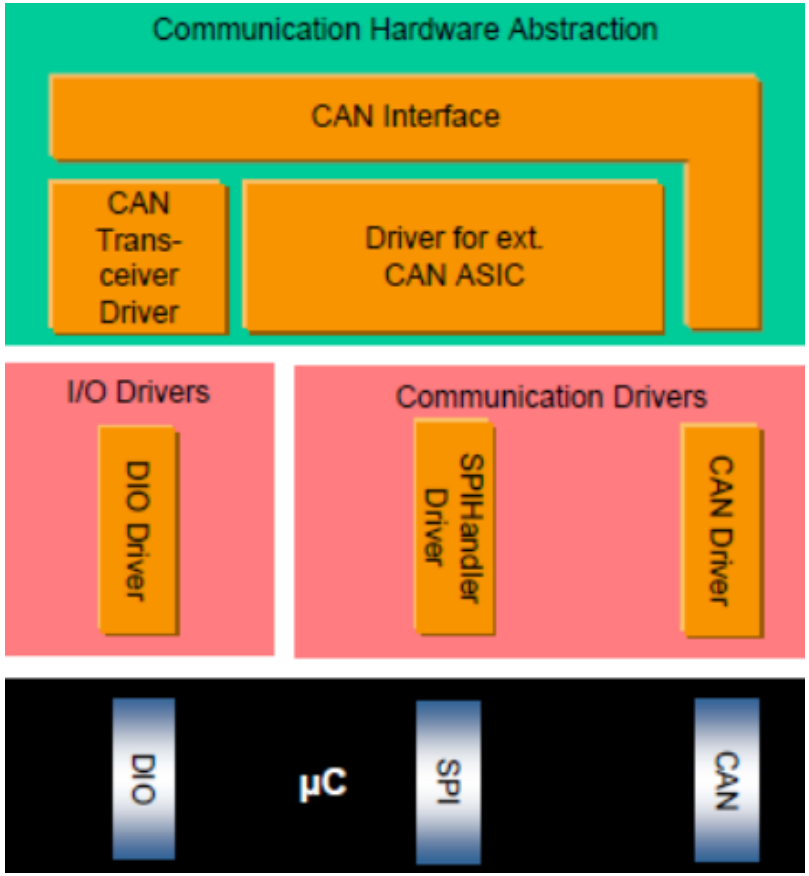


COM硬件抽象：

COM硬件抽象将微控制器、板上的所有通信通道进行了封装。

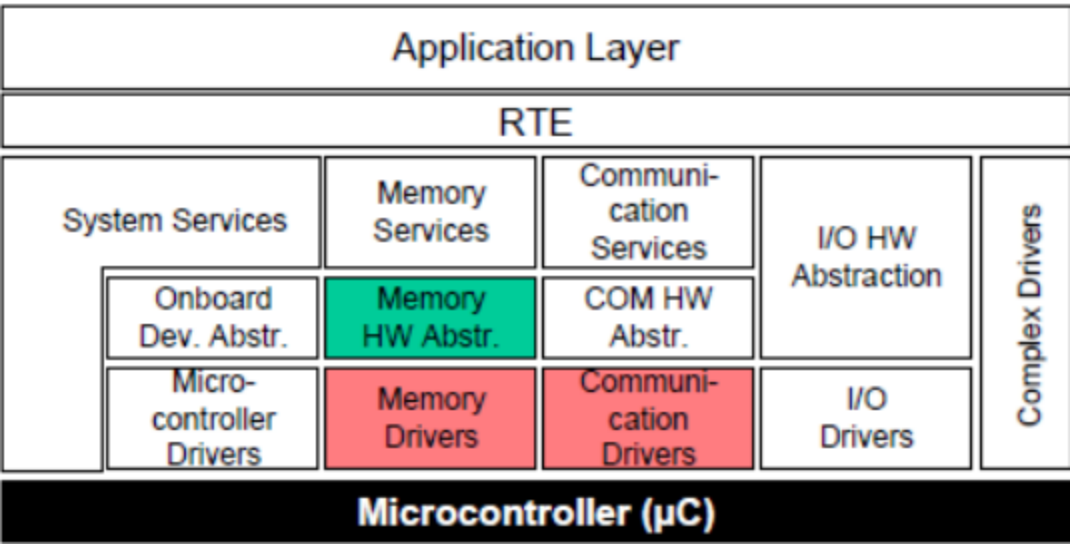
将LIN、CAN、FlexRay等通信方式都进行了抽象定义。

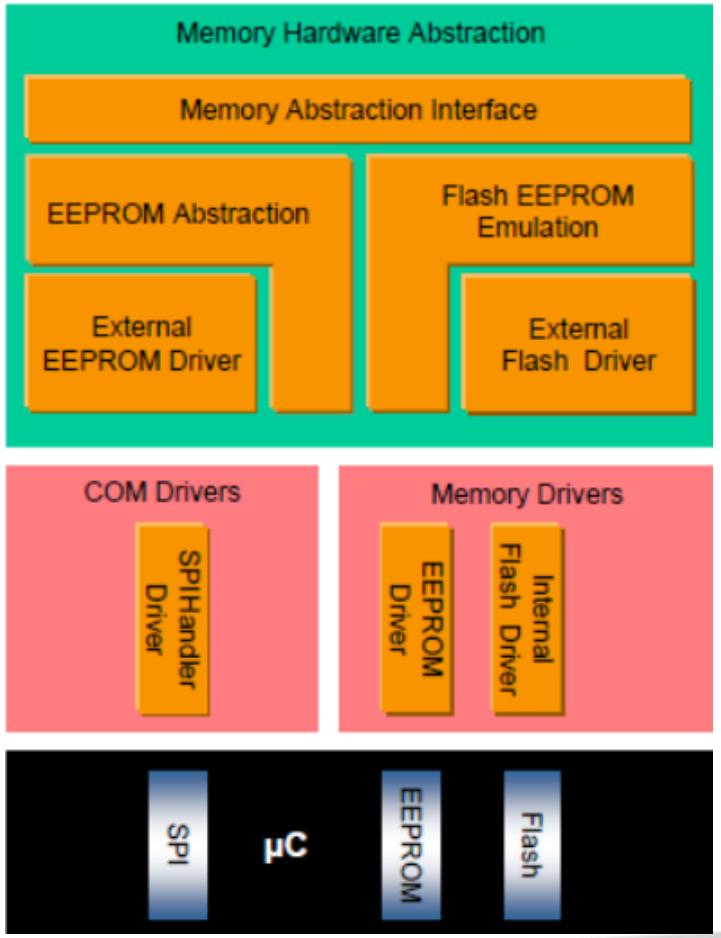




Memory硬件抽象：

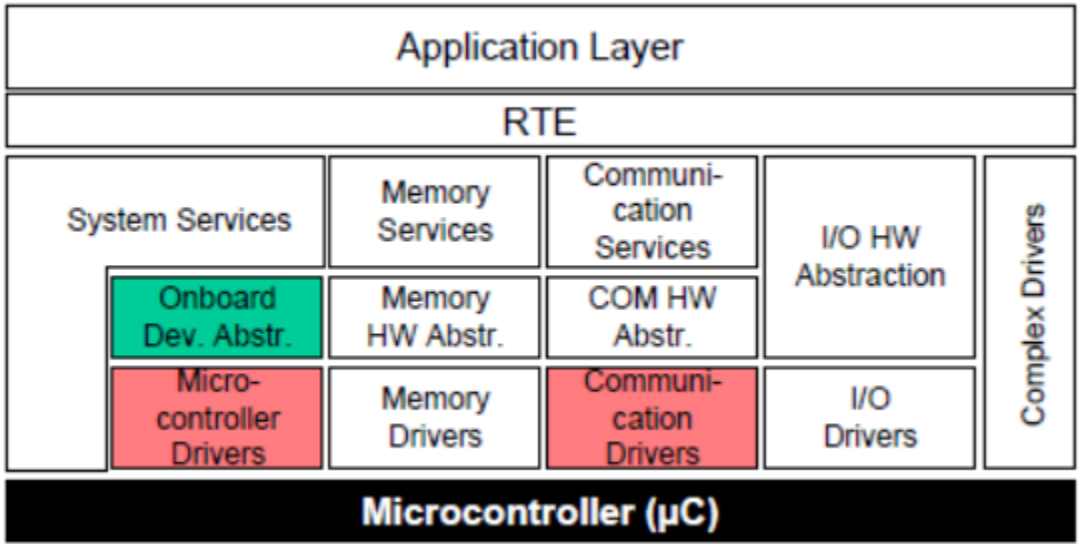
将片内、板上的内存资源进行了统一封装，比如对片内EEPROM和片外的EEPROM都提供了统一的访问机制。

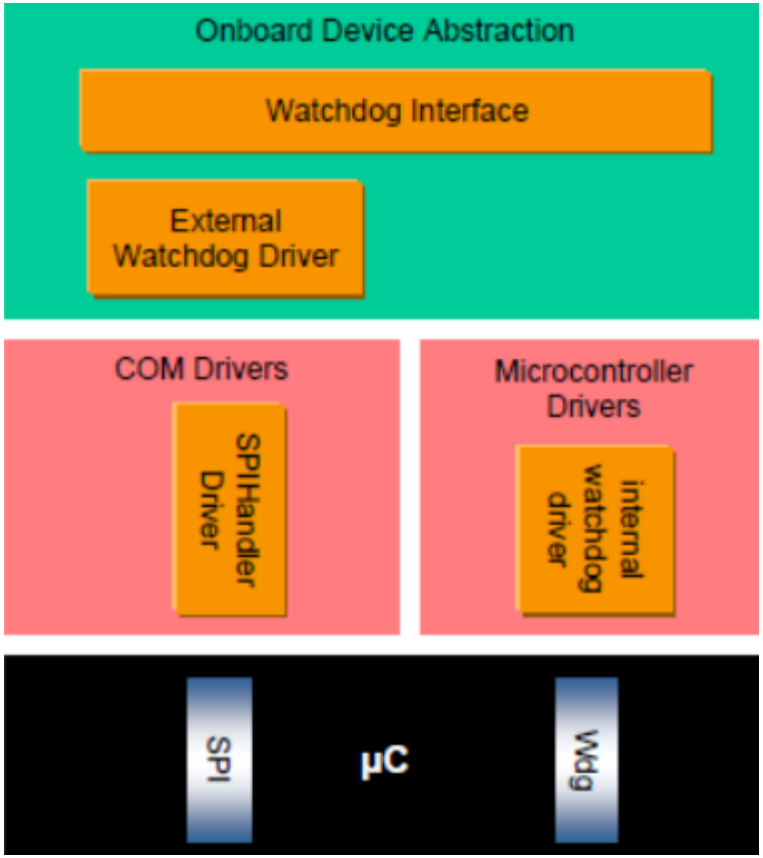




Onboard Device Abstraction：

将ECU硬件上特殊的外设（即不是用于传感，也不用于执行的）进行封装，比如Watchdog。





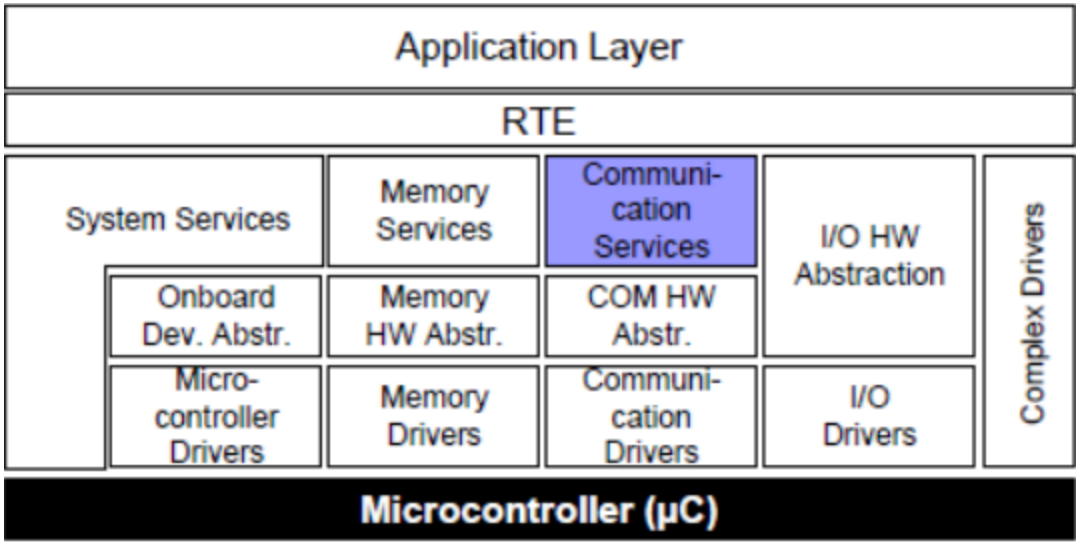
通信服务：

通信服务封转了CAN、LIN、FlexRay等通信接口：

提供统一的总线通信接口

提供统一的网络网络管理服务

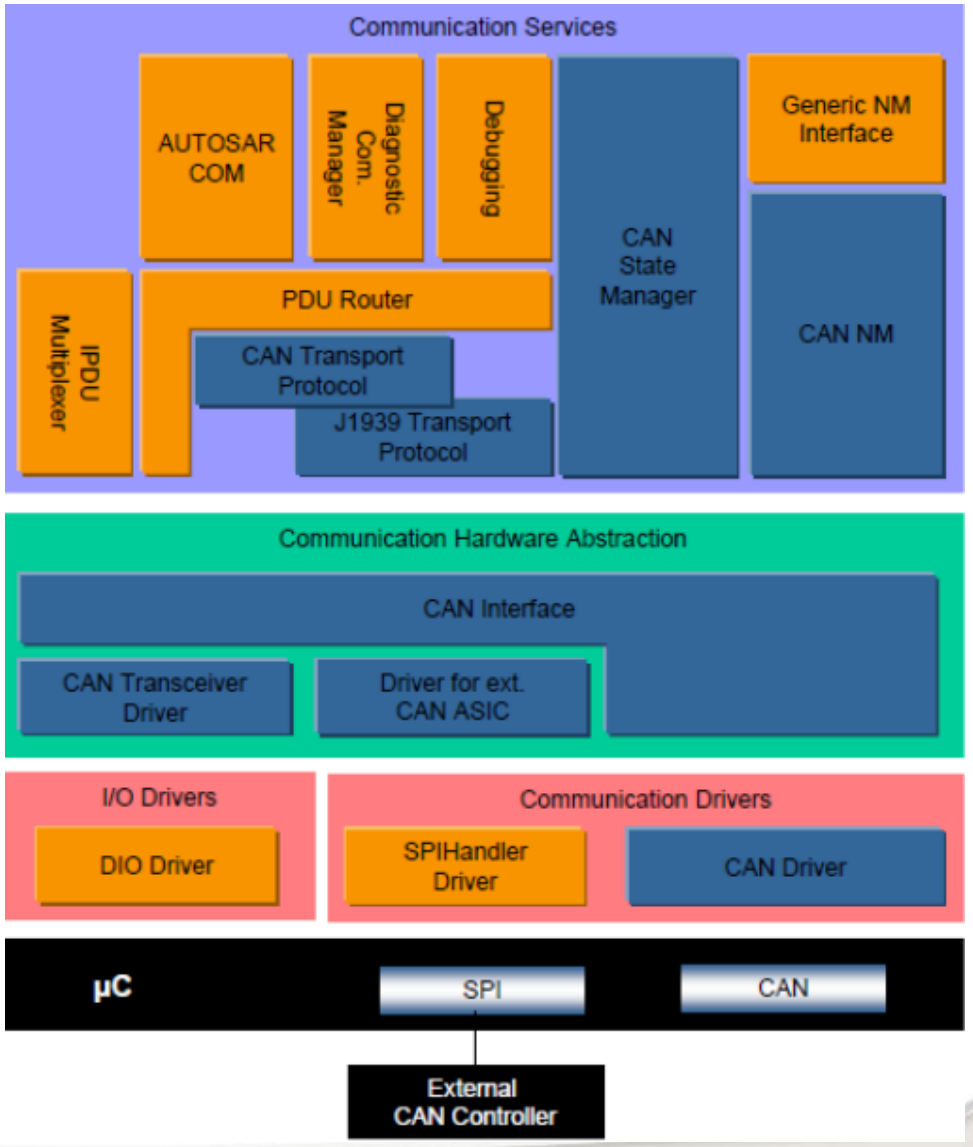
提供统一的诊断通信接口



通信服务-CAN：

针对CAN的通信服务封装了具体的协议、消息属性，提供了统一的接口供应用层调用。

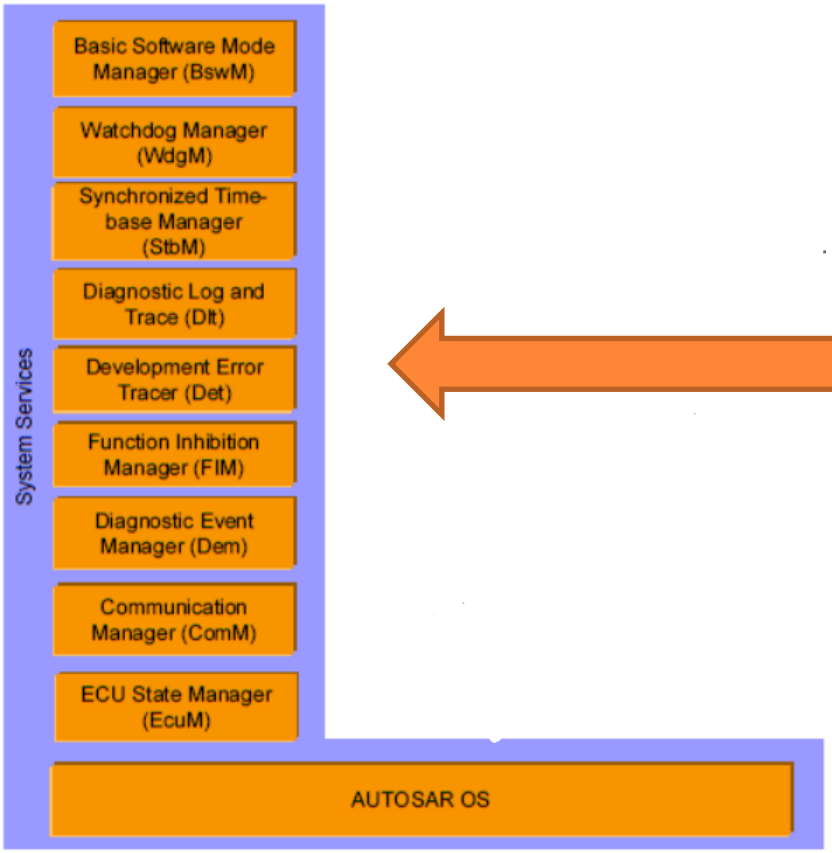
两种传输协议：J1939 TP、CanTP。

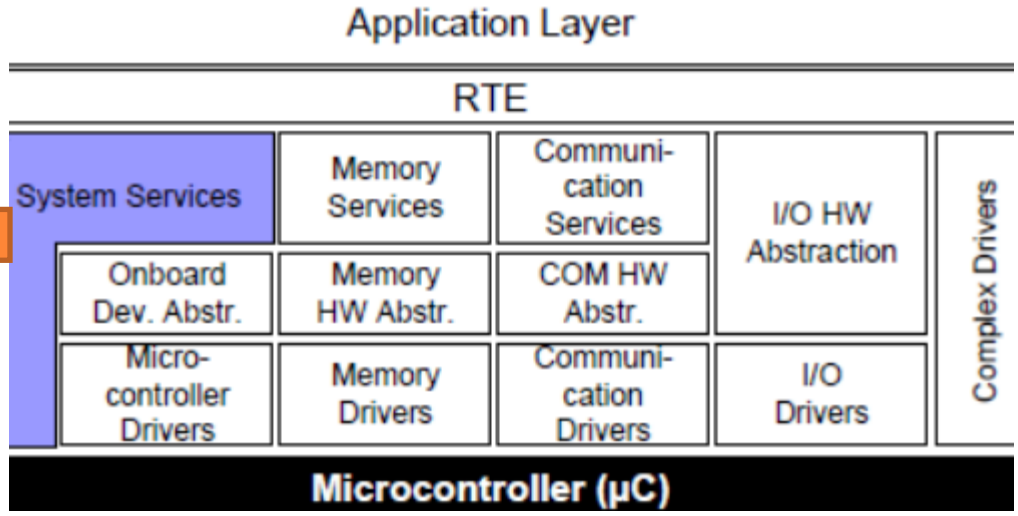


系统服务：

提供RTOS服务，包括中断管理、资源管理、任务管理等。

提供功能禁止管理、通信管理、ECU状态管理、看门狗管理、同步时钟管理、基本软件模式管理等服务。





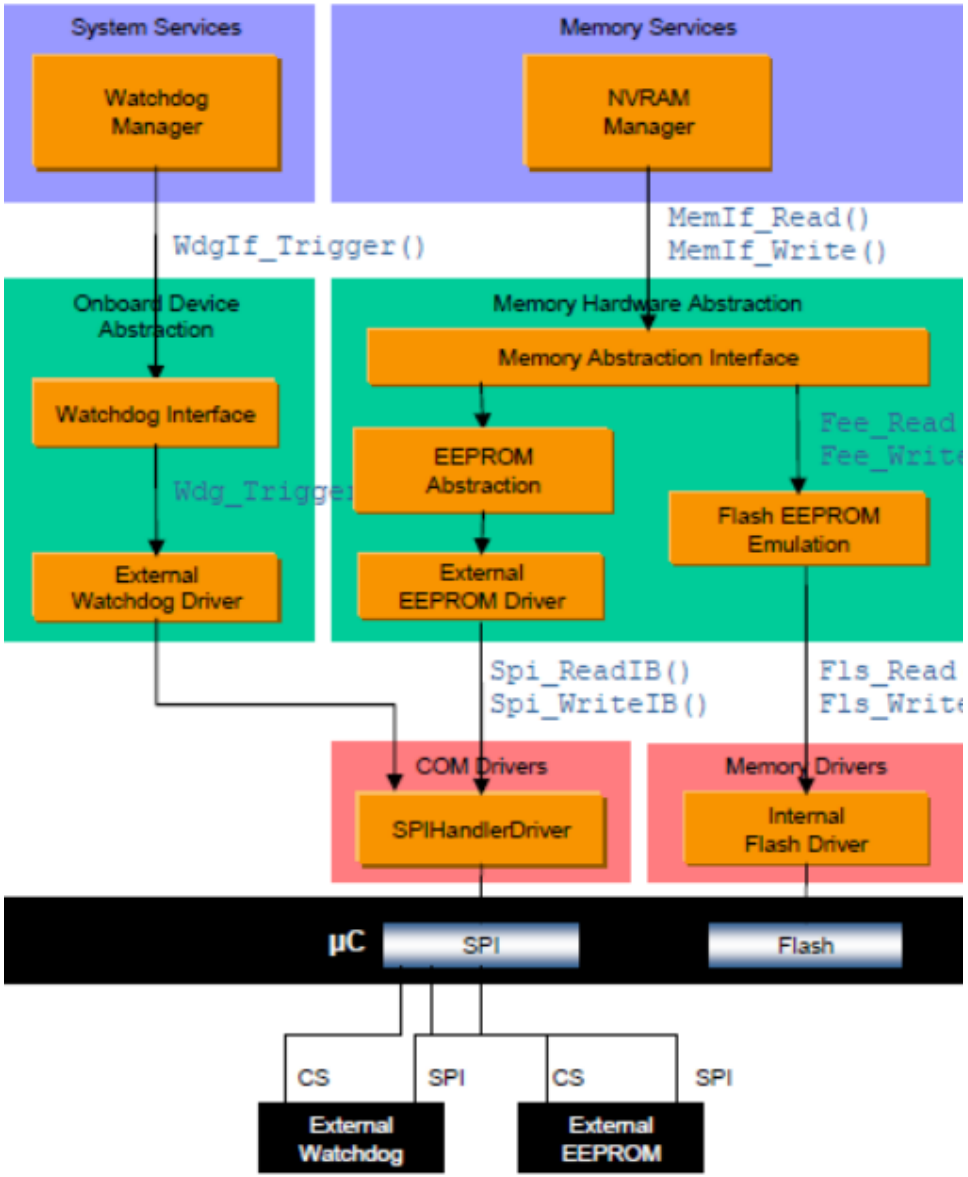
BSW示例：

假设硬件条件如下：

ECU包含外部EEPROM和外部Watchdog，都通过SPI来与微控制器连接；

SPIHandlerDriver控制SPI的并发访问，其中EEPROM的访问优先级更高；

微控制器也含有EEPROM，可以与外部EEPROM同步使用。



- 六、SmartSAR studio使用总结

  1.方法概述

  2.系统设计

  3.系统配置

  4. ECU配置

  5. 执行文件生成

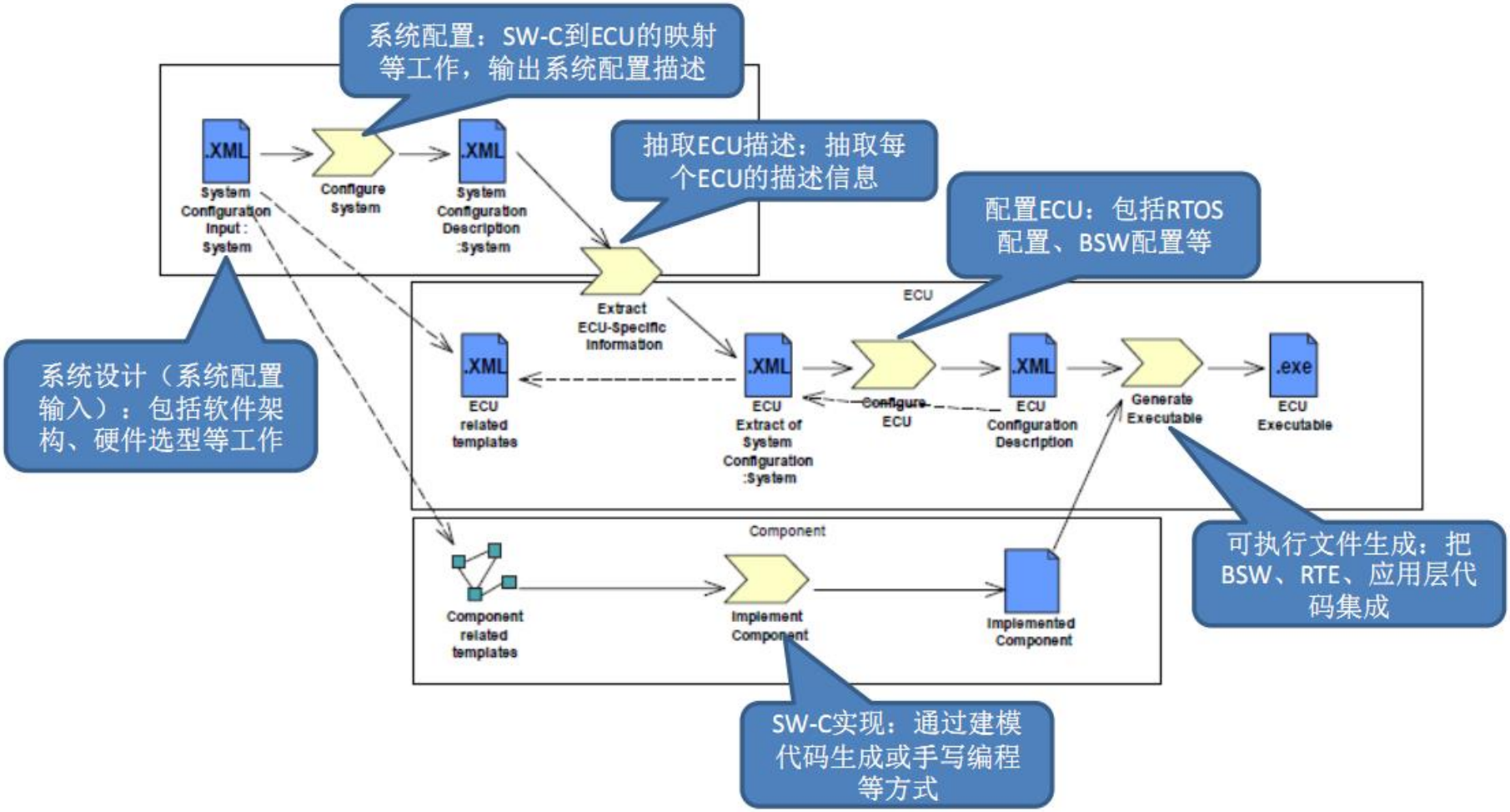
1.方法概述

应用层由各个SW-C设计组成

在VFB上可以验证SW-C间的接口一致性

利用工具进行系统配置、ECU配置、代码生成

支持整车电控系统设计


系统配置：SW-C到ECU的映射等工作，输出系统配置描述
抽取ECU描述：抽取每个ECU的描述信息
配置ECU：包括RTOS配置、BSW配置等
系统设计（系统配置输入）：包括软件架构、硬件选型等工作
可执行文件生成：把BSW、RTE、应用层代码集成
SW-C实现：通过建模代码生成或手写编程等方式
.XML
System Configuration Input : System
Configure System
.XML
System Configuration Description :System
Extract ECU-Specific Information
ECU
.XML
ECU related templates
.XML
ECU Extract of System Configuration :System
Configure ECU
.XML
ECU Configuration Description
Generate Executable
.exe
ECU Executable
Component
Component related templates
Implement Component
Implemented Component

2.系统设计

系统配置输入：

软件架构设计：设计软件组件SW-C，包括data types、ports、interfaces等

收集ECU资源：处理器、内存、外设、执行器、传感器等规格

指定系统约束：总线速率、总线拓扑等约束

系统设计之软件架构：

SW-C的设计和新建

SW-C的输入输出的定义（Port、Interface）

SW-C之间的关系绑定（Connector）

系统设计之SW-C实现：

通过Simulink等工具建模

系统设计SW-C描述最后导出arxml文件

3. 系统配置

硬件约束输入

总线的硬件拓扑

总线约束输入

通信矩阵设计

SW-C映射

SW-C与硬件映射

4.ECU配置

MCU基本配置
信号引脚的配置
模拟输入信号的配置（ADC）
模拟输出信号的配置（PWM）
总线信号的配置（CAN，SPI）
RTOS等模块的配置
RTE的配置

5.生成可执行文件

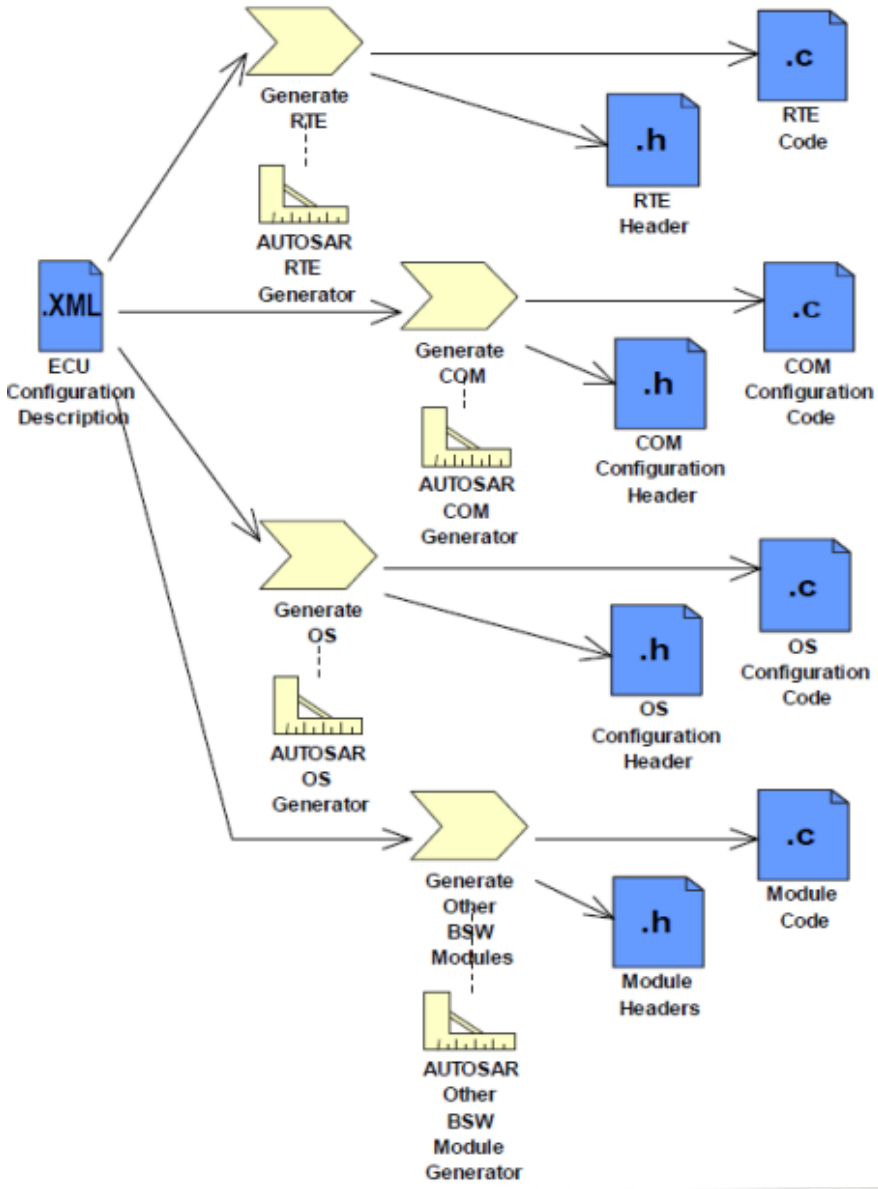
生成代码：

生成RTE

生成COM的应用

生成OS的应用

生成BSW的应用



生成可执行文件：

链接以下目标代码：

编译后的SW-C

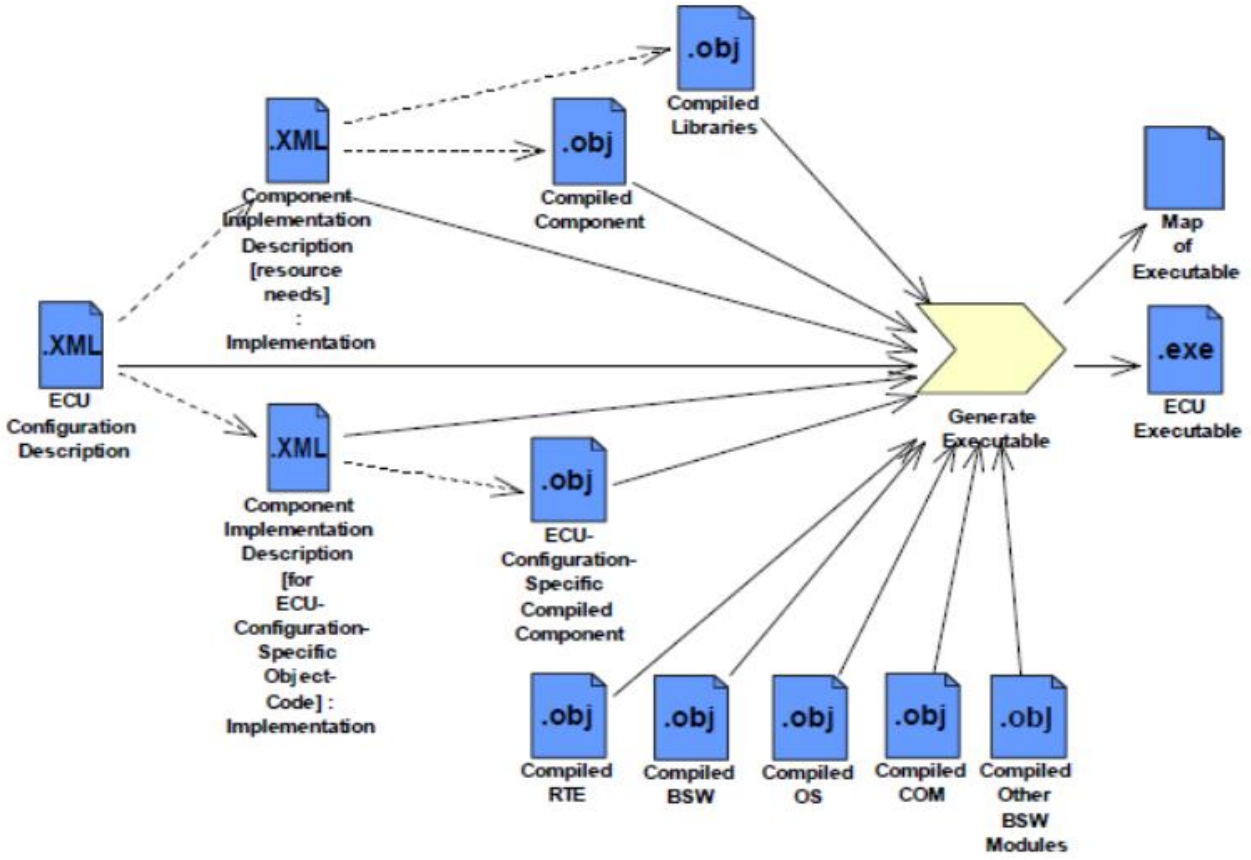
SW-C的相关库

编译后的RTE

编译后的BSW

编译后的COM

编译后的OS



谢谢大家！