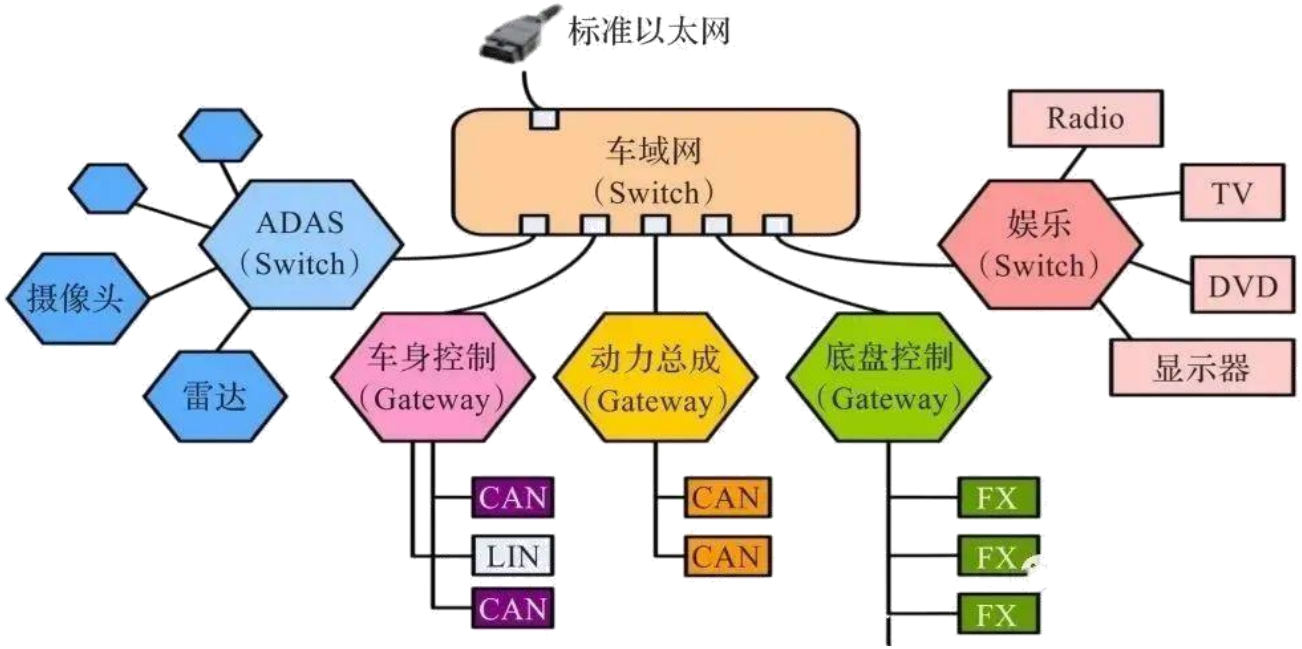
车载上的跨系统融合

熊谱翔 @ RT-Thread

目录

汽车ECU架构情况

- 现代汽车整体是一个复杂的架构，包含上百个ECU控制器
- 娱乐域
- 辅助驾驶域
- 车身/动力/底盘控制



车上ECU系统

CP AUTOSAR - Classic Platform AUTOSAR

- SWC – 应用软件层
- RTE – 实时运行环境
- BSW – 基础软件层
  - MCAL – 微控制器抽象层
- MCU – 微控制器

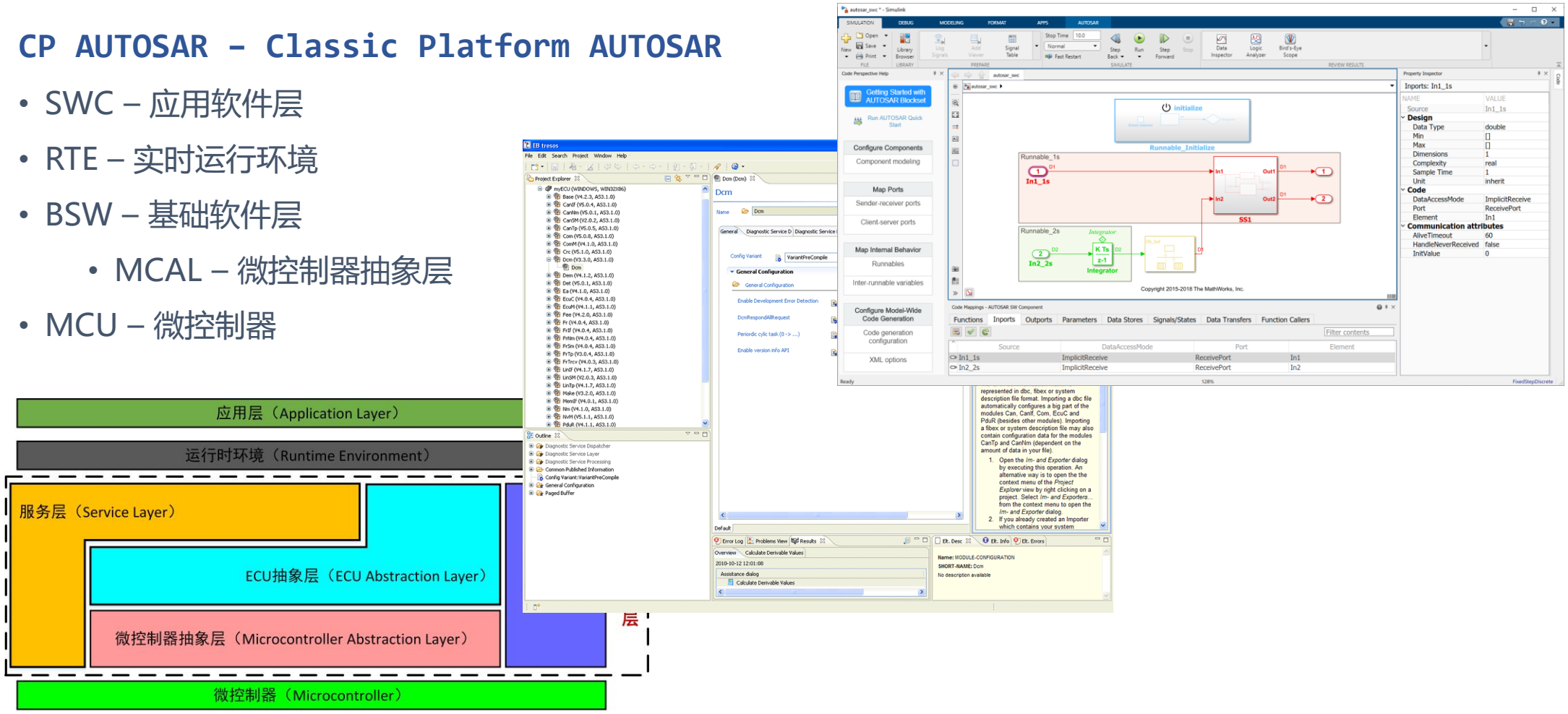


图1 AUTOSAR体系架构

又爱又恨的CP AUTOSAR

好的一面

- 依赖于工具的代码生成，安全性非常高（验证过的机器，及其不容易出错）；
- 摆脱繁杂的代码编写，低代码开发；

不好的一面

- 冗余，资源占用过大
- 存在约束
    - 在框框下完成工作；
    - 没有太多的可发挥的空间，也限制了想象力；
- 市场被几家厂商垄断
    - 相互并不完全兼容；
    - 对新芯片支持没有那么迅速，特别在缺芯的市场环境中；

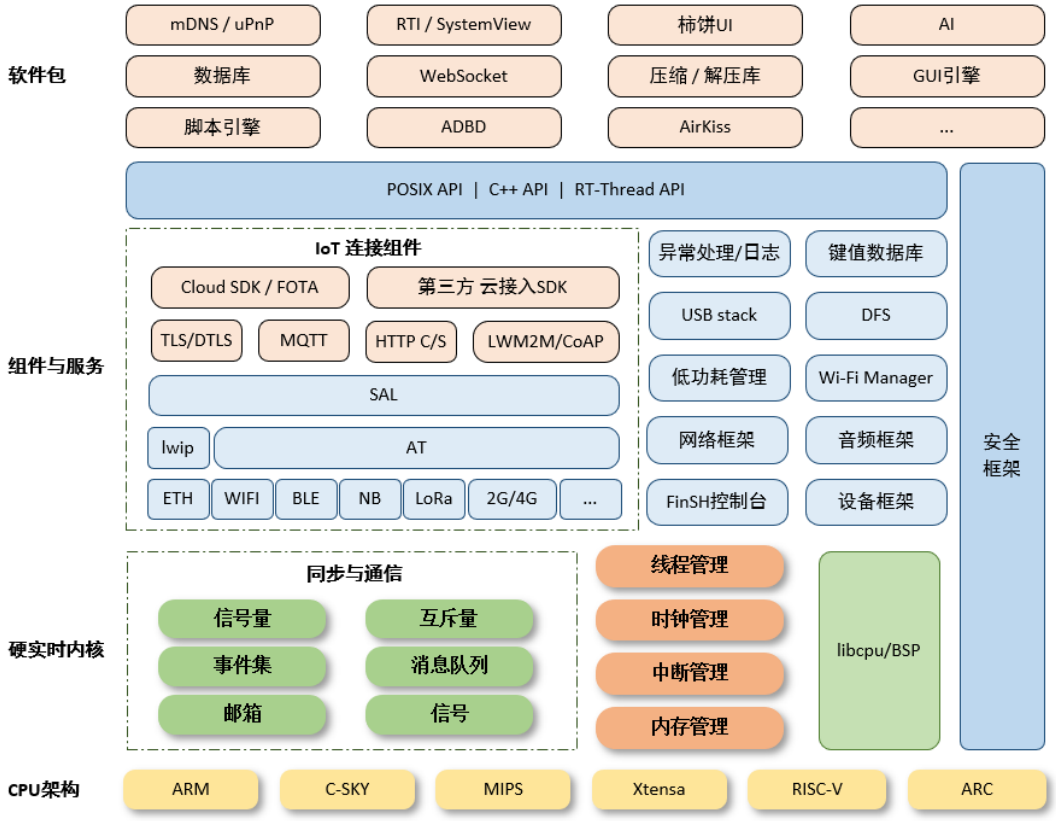
RT-Thread 硬实时嵌入式操作系统

开源开放

- 开放平台，软件包生态；
- 支持众多芯片，开发板；

长期沉淀

- 积木式架构，裁剪由心，可大可小
- 支持静态内存方式执行，支持MPU内存隔离执行；
- 广泛应用于工业，电力，轨道，航天等领域；



传统 + 创新

- 硬实时操作系统，抢占式多任务调度；
- 非常全的POSIX环境；
- 丰富的软件包生态，涉及到工业，AI，系统工具，脚本等方方面面。

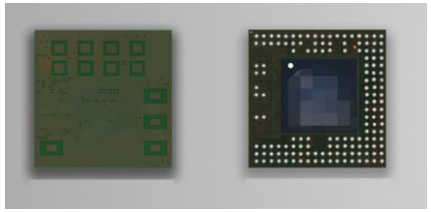
最全的工具链支持

- 最不挑工具链的操作系统；
- GNU GCC，llvm，armcc/armclang，iar，tasking，CCS
- All in One集成开发环境
  - 开发、配置、调试

用于ARC芯片上的技术参数情况

- ARC EM6架构的毫米波雷达芯片移植；
- 300MHz处理器
- 支持周期触发方式调度，调度表

	邮箱切换	信号量切换	直接唤醒切换	中断响应延时
最小时间	0.816	0.703	0.626	0.316
最大时间	1.903	1.126	1.380	1.410
平均时间	0.816	0.703	0.626	0.326



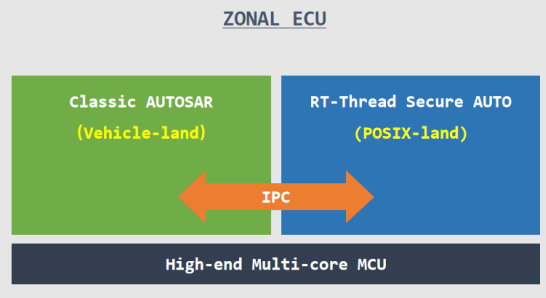
小型资源占用

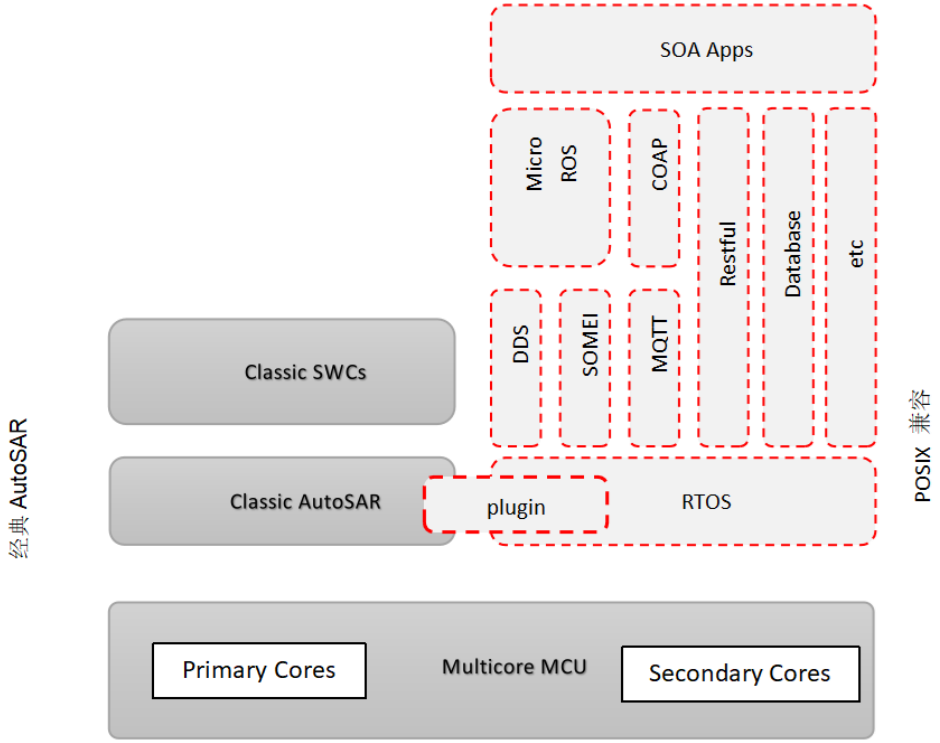
```
arc-elf32-size rtthread_embarc.elf
   text    data     bss     dec     hex filename
  92612    3648   10372  106632   1a088 rtthread_embarc.elf
scons: done building targets.
```

POSIX RTOS + CP AUTOSAR

Gemini OS:　（和蓝马舱行智能科技（上海）有限公司合作）

- POSIX RTOS + CP AUTOSAR的相互补充、结合；
- 在高端多核MCU上把多核分区成主核分区和从核分区；





CP AUTOSAR

- 用于功能要求较高的车身电子等功能；
- 提供CDD（Complex Device Driver）做为插件，和POSIX RTOS进行通信；

POSIX RTOS

- 面向新的SOA业务，提供POSIX环境；
- 使用RPMsg方式进行通信，和CP互联互通起来；
- 相互隔离，不相互Touch内存，保证安全性；

提升以太网性能

MCAL的以太网性能偏弱

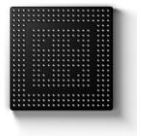
- 对于网络报文的处理有比较多的冗余，性能不佳

RT-Thread系统下

- 可以很好的补充以太网性能及上层应用；
  - 高精度时间同步
  - VLAN等
- 在实时性上依然可以得到保证（硬实时系统），适用于一些新的SOA场景。

MPU SoC

- 从影音娱乐，到辅助驾驶，自动驾驶，中央电脑
  - 人机体验
  - 智能操控
  - 整车中央大脑



- GPU，MIPI/HDMI
- Camera

- 几十 – 上百 Tops

- AI，MIPI/HDMI
- Camera, ISP
- MCU

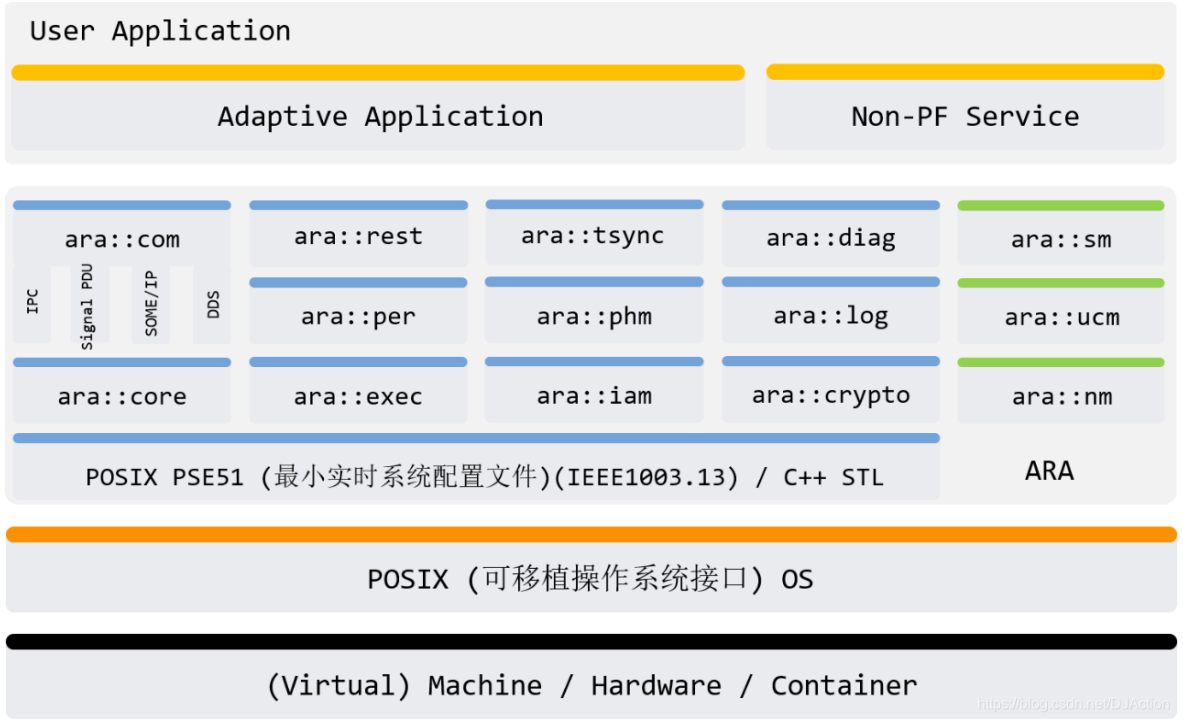
- 上千Tops

- AI，GPU, MIPI/HDMI
- Camera, ISP
- MCU

AP AUTOSAR

Adaptive Platform AUTOSAR

- 面向高性能计算，MPU高性能处理器平台；
- 底层基于POSIX操作系统，最小PSE51子集实现；
- 遵循面向服务的架构设计理念，实现Service Interface；
- 主要使用C++语言开发，C++11/14，也包括C++17
- 更多是在RAM中执行；



功能安全的三层架构

Level 1：功能层

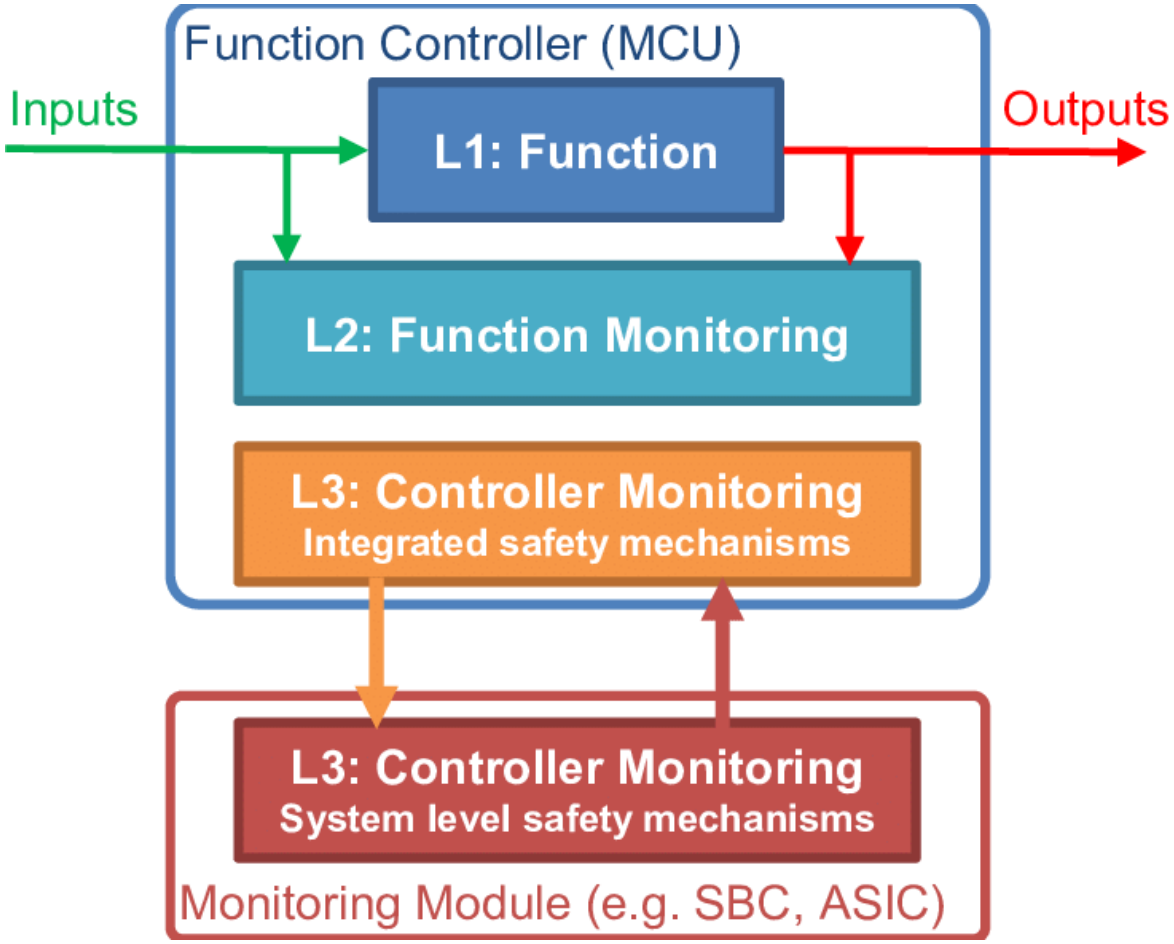
- 控制器相关功能逻辑实现

Level 2：功能监控层

- 对功能逻辑实现的缺陷进行监控

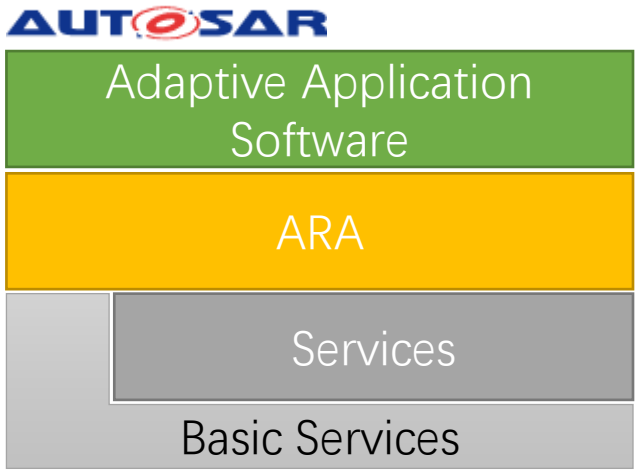
Level 3：控制器监控层

- 对整体控制器进行独立的监控
- 独立的SBC或MCU

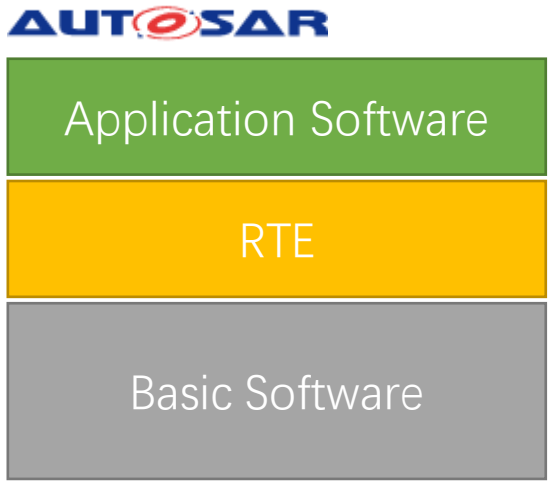


AP + CP

Adaptive Platform

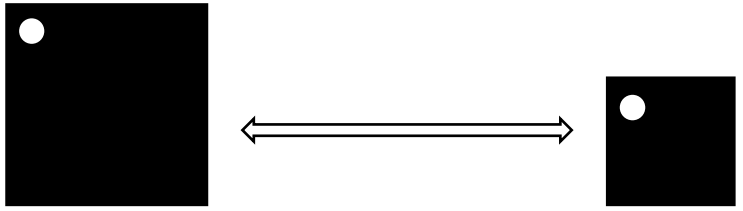


Classic Platform



MPU高算力芯片

- 使用以太网方式进行通信；
- 面向服务的架构；



ETH

CAN/FD
LIN

MCU芯片

- 对MPU进行安全监控；
- 车身控制；

MPU上的系统

- 大家的期待：
  - 性能能够像Linux一样的优秀，同时有优异的实时性、功能安全特性。

Linux

- 优秀的生态，特别是在Android上
- 软实时，缺乏功能安全认证

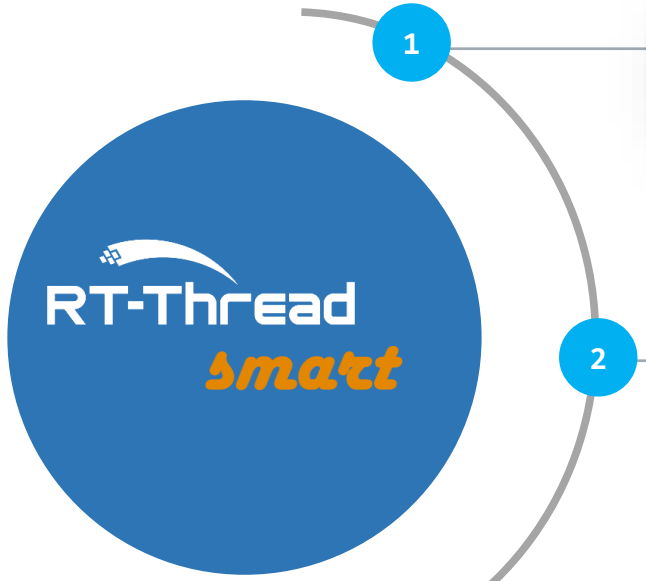
QNX

- 微内核操作系统
- 内核、核心经过功能安全认证

RT-Thread smart

- 混合微内核架构
- 具备 ASIL-D 功能安全认证

RT-Thread Smart：实时、安全、POSIX生态



1

高实时性

延续RT-Thread RTOS的特点，具备高实时性；

系统实时性，中断延迟低于3us；

2

安全内核及系统服务

用户态驱动、用户态系统服务，做到内存安全隔离，相互不影响；

统一的系统服务管理器，可对服务进行监控管理；

3

POSIX生态

完整的POSIX兼容能力，通过Open POSIX Test suite兼容性测试

无缝移植Linux/Unix等POSIX生态应用程序

vmRT-Thread：虚拟化系统扩展，面向不同的功能区

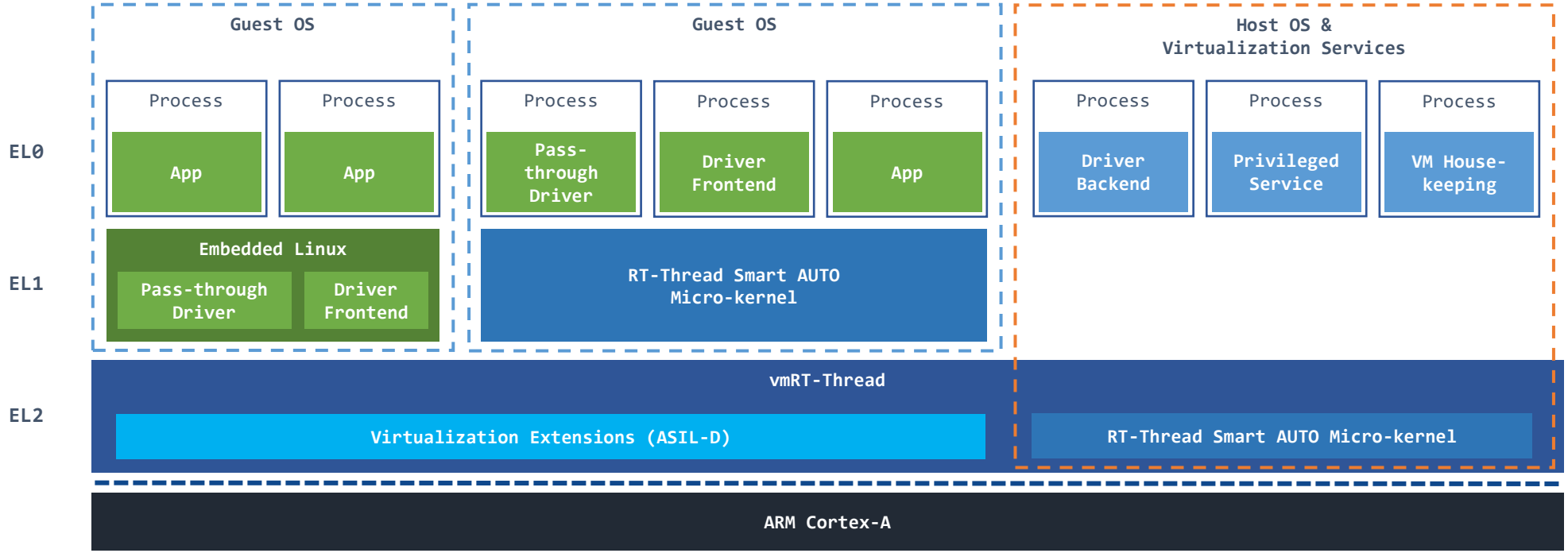
Virtualization Extensions (VirtExt)

- 基于CPU微架构/指令集的虚拟化扩展，以ARM为例如：
  - hypercall 实现
  - GIC，Stage2-MMU，SMMU 配置等
- 运行于 EL2，ASIL-D

Virtualization Services (VirtSrv)

- 虚拟机管理（启动/关闭/监控）
- 用于外设半虚拟化的驱动后端
- 特权服务（用户自定义）
- 运行于 EL0，部分功能安全相关组件 ASIL-D

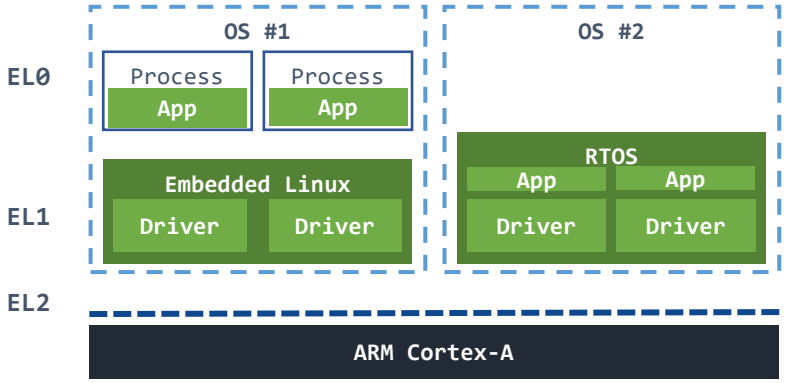
- **Host OS**
- 微内核实时操作系统
- 用于运行/调度/管理所有的 VirtSrv 和自定义任务
- 不同 VirtSrv 的功能安全等级不同
- 内核运行于 EL2，ASIL-D



AMP vs 虚拟化系统

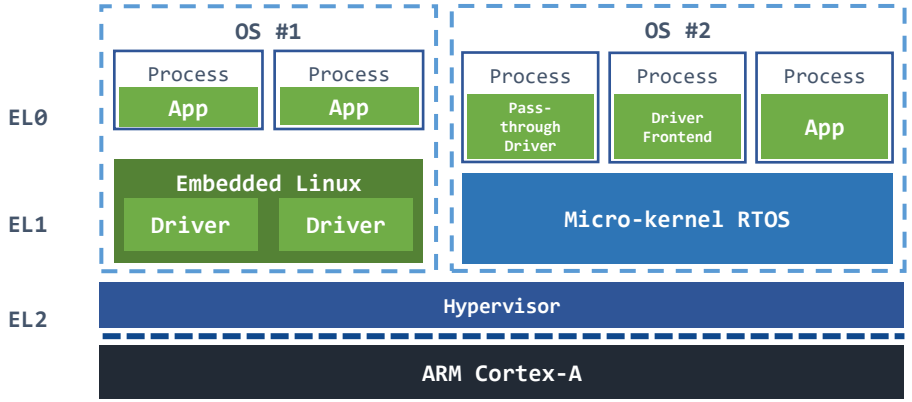
AMP - “硬隔离”

- 在同构多核上运行多种操作系统，例如AArch64多核；
- 直接硬件操作，效率高：
  - 性能，吞吐量高，实时性高
- 并不能实现完全的硬件隔离
  - 内存需要到更底层进行监管
  - 共用的硬件外设操作需要非常小心
    - 例如GIC中断控制器



虚拟化系统

- 在多核上运行多种操作系统；
  - Linux，RTOS，Android并行执行
- 硬件隔离好
  - 充分的运用硬件特性，在更底层进行隔离
  - 虚拟化共用的硬件外设；
- 实时性，性能会受一定影响
  - 实时性和中断密切相关，受中断注入的影响；
  - 虚拟的/半虚拟化的外设性能偏低；
  - 复杂外设的虚拟化难度会比较高，特别还和性能相关时；



谢谢

关注RTThread公众号获得更多资讯