多核处理器与实时操作系统 - 概述、原理和应用实例

何小庆 何灵渊

Allan@esbf.org

讲座的内容

- 图书内容介绍-嵌入式实时操作系统-理论基础
- 多核处理器芯片硬软件架构
- RTOS的任务调度与资源共享
- 多核处理器实时操作系统应用
- 在树莓派Pico上通过FreeRTOS实现对称多处理

《嵌入式实时操作系统 – 理论基础》介绍(1)

- 本书内容 (13章)
  - 实时操作系统基础
  - 调度——概念和实现
  - 使用互斥机制控制资源共享
  - 资源共享和争用问题
  - 任务间通信
  - 存储的使用和管理
  - 多处理器系统
  - 分布式系统
  - 调度策略的分析
  - 操作系统：基本结构和功能
  - RTOS的性能和基准测试
  - 多任务软件的测试和调试
  - 在关键系统中使用RTOS

何小庆老师图书资料下载

后面“图书” 指《嵌入式实时操纵系统 -理论基础》

《嵌入式实时操作系统－理论基础》介绍（2）

- 作者简介
  - Jim Cooling　博士，在嵌入式实时操作系统领域拥有多年经验，出版了多本著作，涵盖如实时操作系统编程、软件设计和软件工程。曾任英国飞机公司飞行控制系统设计师；Marconi Radar Systems 电子电路和系统设计师；英国海军电子控制系统项目经理；英国拉夫堡大学研究员和高级讲师。
- 译者简介
  - 何小庆　张爱华　何灵渊　付元斌
  - 团队有多年嵌入式软件开发经验，2012以来已出版5本译作，致力于推动嵌入式系统产业发展并支撑嵌入式系统学术研究发展。

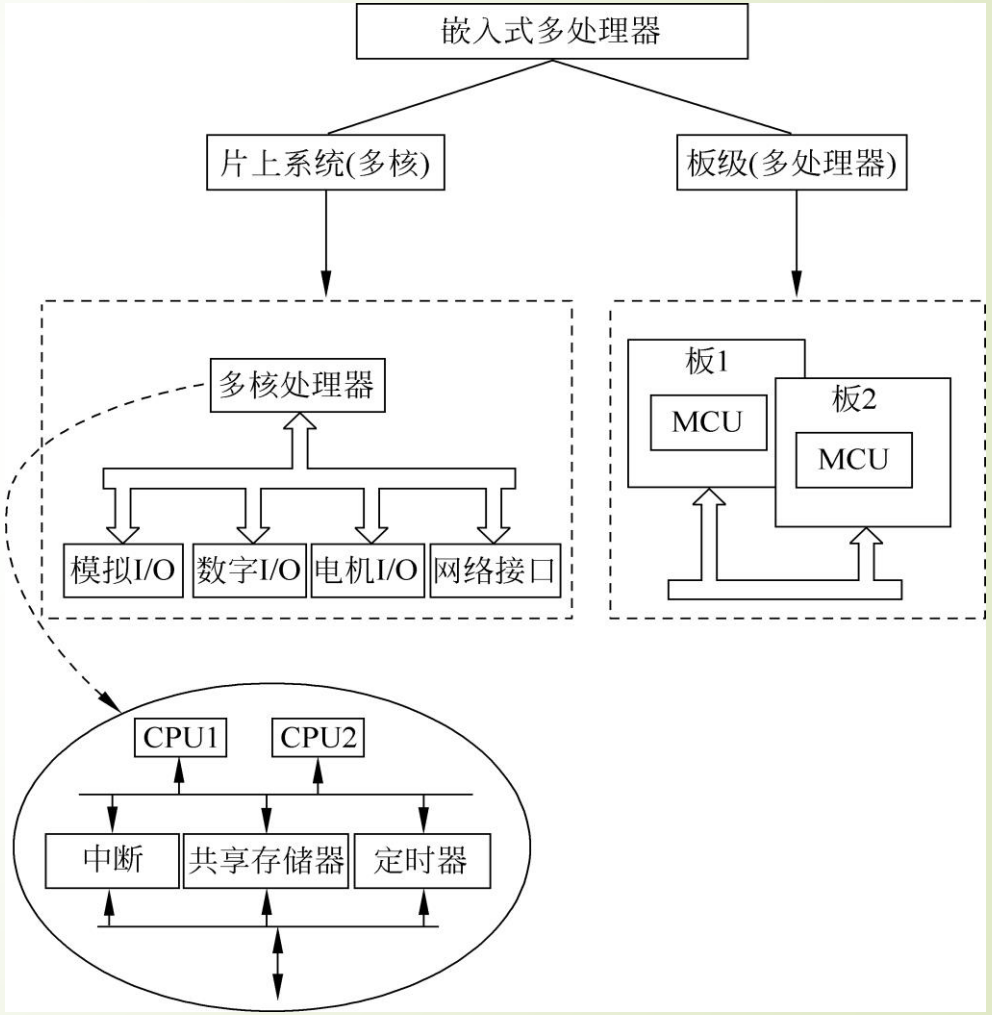
何小庆老师图书资料下载

深入剖析理论知识 联系实际系统工程 关注安全关键系统

多核处理器芯片硬软件架构

微处理器、多处理器和多核处理器

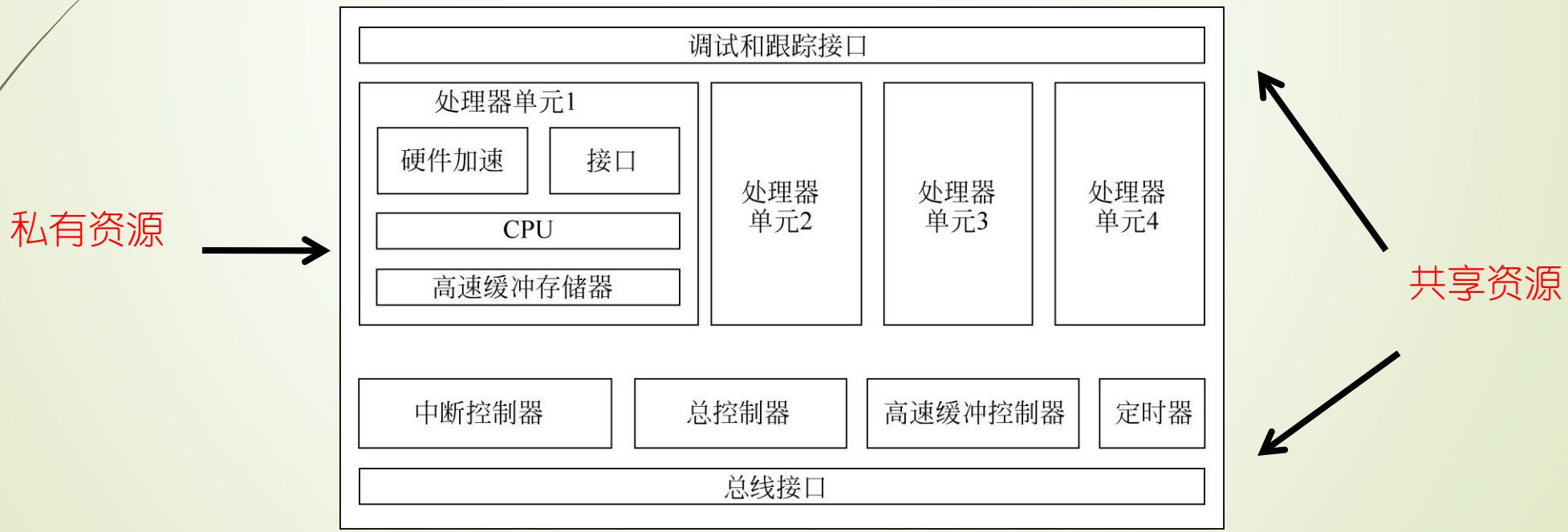
- 1971年， Intel公司设计出第一款微处理器 - 4004，很快Intel又推出了8位的8008处理器和16位的8086处理器。
- 8086芯片单核CPU，随着需求的提高和功耗问题出现，就有在一个芯片上建造两个或者多个核。比如 Intel i5（14代）6+8 核，单核 5.3G.
- 多核CPU具有更高的计算密度和更强的并行处理能力并且尺寸还很小，多核的发展正在改变计算的未来，多核处理器将适合混合关键系统。



本文在后面的讨论中：多处理器、多核处理器、多核、多核芯片 不加以区别

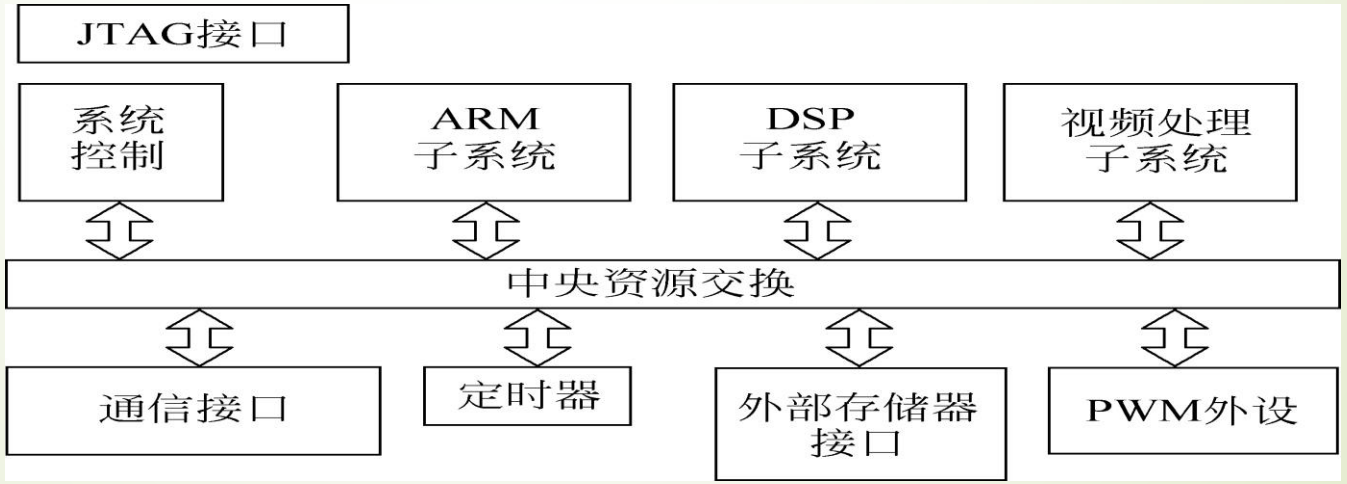
多核处理器的同构和异构 （1）

- 从硬件的角度看，多核处理器可以分为同构和异构两种架构。
- 同构
  - 所有的CPU的核心架构都一样，那么称为同构。例如，瑞芯微RK3568、飞思卡尔的I.MX6D，它们有两个或者以上架构相同的ARM Cortex-A核 都属于同构。
  - 比如 RK3568 四核64位Cortex-A55 主频最高2.0GHz,内置1.0TOPS算力NPU



多核处理器的同构和异构（2）

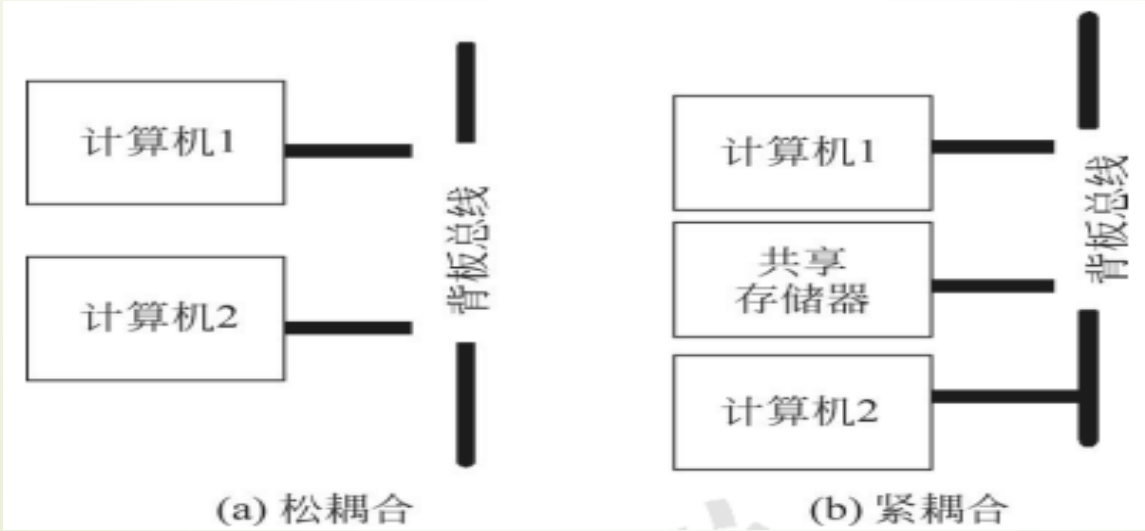
- 异构
  - CPU核心架构有不一样的，就称为异构。例如 STM32MP157，有两个ARM Cortex-A7核和一个ARM Cortex-M4核，Xilinx的ZYNQ7000系列，有两个ARM Cortex-A9核和FPGA（可配置MicroBlade），TI TMS320DM8127有一个DSP C674x核和一个ARM Cortex-A8核，这些处理器有不一样架构的CPU核，所以都属于异构。
  - 异构多核CPU 可根据负载，平衡算力的使用，典型是手机大小核配置，以及特殊需求（比如实时性计算需求可由M/R核完成，人机界面由A核完成）等。



多处理器系统通信结构

参考图书第7章：多处理器系统

- 嵌入式多机系统有松耦合和紧耦合两种形式。
- 在松耦合系统（a）中，计算机之间的通信是通过网络传输技术进行的，这样的好处是彼此间的依赖最少。这种技术可更方便地构建一个多机系统。缺点是计算机之间通信时间会比较长，这样，性能就成为松耦合系统的一个瓶颈。
- 在紧耦合多机系统（b）中，计算机间的通信使用共享存储器。从设计角度看，这样的系统很类似于分布式任务结构，任务间的通信通过邮箱机制进行。

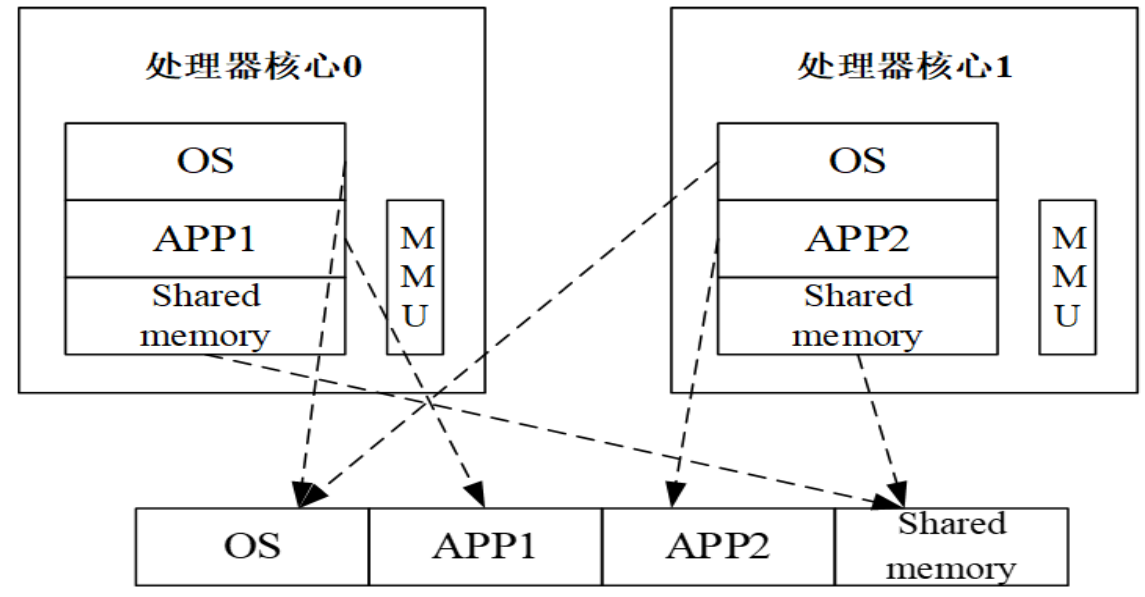


对称多处理架构（1）

- **从软件的角度看：**多核处理器平台的系统架构有：SMP（Symmetric Multi-Processing，对称多处理）架构、AMP（Asymmetric Multi-Processing，非对称多处理）架构和BMP（bound multi-processing，绑定多处理）架构。
- SMP架构是指只有一个操作系统（OS）实例运行在多个CPU上，一个OS同等地管理各个CPU内核，为各个内核分配工作负载，系统中所有的内核平等地访问内存资源和外设资源。
- 应用的需求，异构处理器的各个内核结构有可能不同。但是，如果一个OS去管理不同的内核，这种情况实现起来比较复杂，所以一般运行在SMP结构下的都是同构处理器。通用OS，如Windows和Linux，嵌入式实时OS如 QNX和SylixOS 等支持SMP结构。

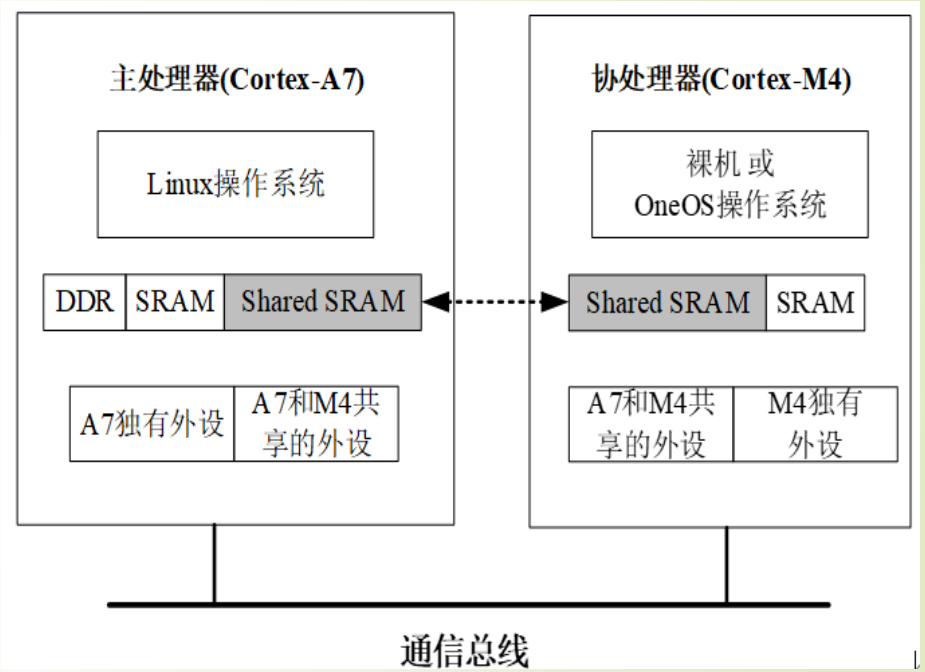
对称多处理架构(2)

- 举一个例子：SMP结构下一个OS负责协调两个处理器，两个处理器共享内存，每个核运行的应用程序（APP1和APP2），通过MMU（内存管理单元）把它们映射到主存的不同物理位置上。



非对称多处理架构(AMP)

- AMP架构是指每个内核运行自己的OS或同一OS的独立实例，或者说不运行OS，如在裸机执行程序。每个内核有自己的内存空间，也可以和其它内核共享部分内存空间，每个核相对独立地运行不同的任务。有一个核为主要核心，它负责控制其它核以及整个系统的运行，而其它核心负责“配合”主核来完成特定的任务。
- 这里主核称为主处理器，其它核心我们就称为协处理器或者远程处理器。这种结构最大的特点在于各个OS都有本身独占的资源，其它资源由用户来指定多个系统共享或者专门分配给某一个系统来使用，系统之间通过共享的内存来完成通信。



STM32MP157 为例

参考《OneOS 进阶开发：异构通信篇》

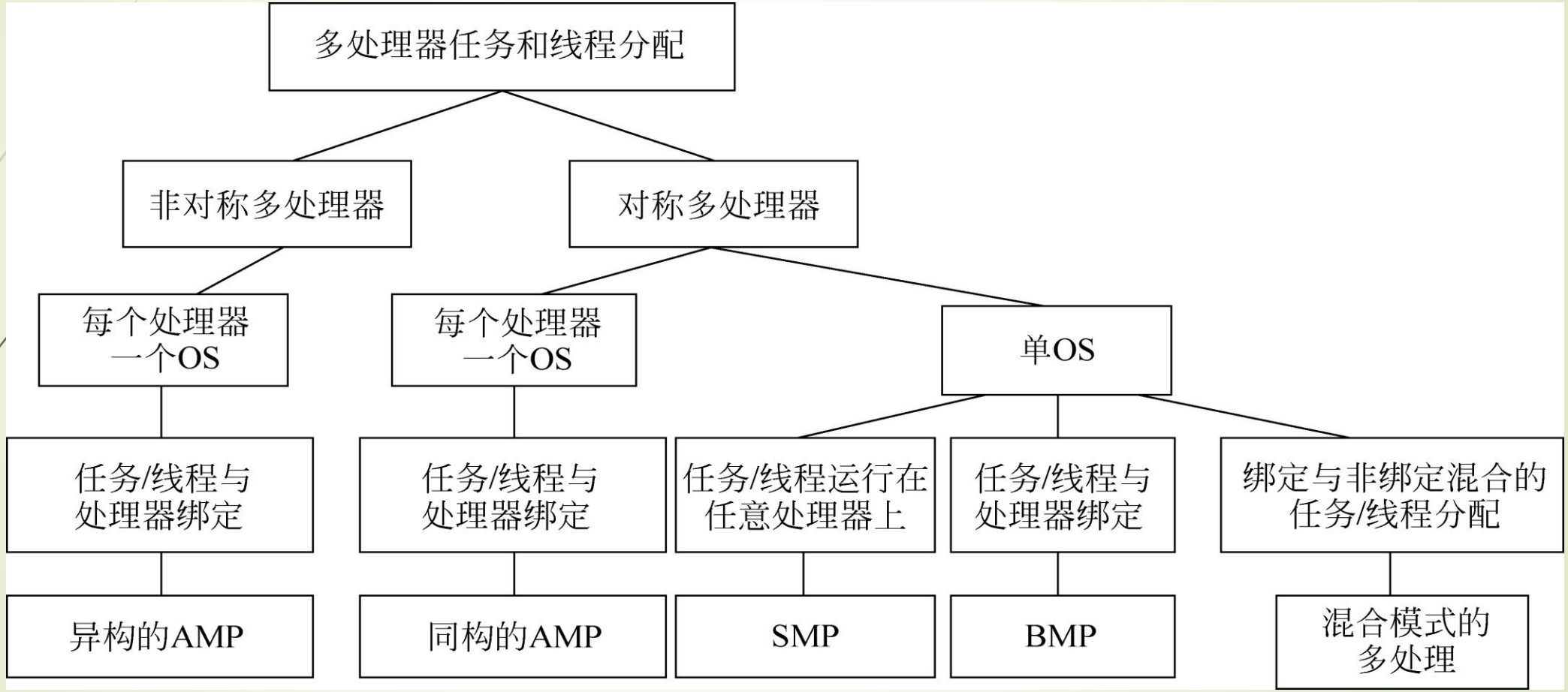
绑定多处理架构(BMP)

- BMP和SMP类似，也是由一个操作系统同时管理所有CPU内核，但是开发者可以指定某个任务在某个核中执行。
- BMP应用更多是在AMP的混合的场景下（hybrid），多个相同架构的内核被配置为一个SMP子系统，另外的不同内核运行的是其它的操作系统，那么从整体来看就是AMP结构了。从逻辑上来分，这个SMP子系统看起来像是一个单核，可以把它包含在这个大的AMP系统中。

解决多核处理器软件问题还可以借助虚拟化技术实现，这次讲座就不展开讨论

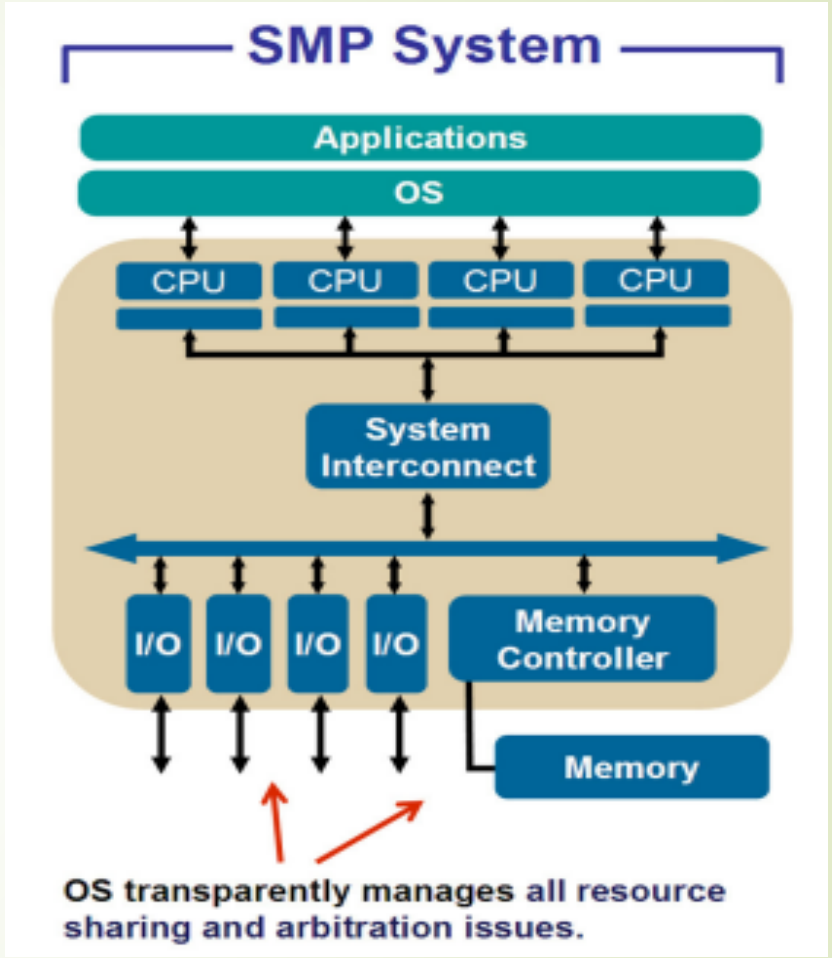
参考：《嵌入式软件开发精解》第10章：多核嵌入式系统

多处理器任务/线程划分总览



对称多处理器SMP的特点

- 作业/任务动态地分配到处理器内核上（运行时）
- 总计算负载分布在每一个内核中。
- 一个操作系统控制所有 CPU 内核上的程序.
- 核间通信和同步不需要特殊的操作系统服务。



图片来自： Case Study — Making a Successful Transition to Multi-Core Processors

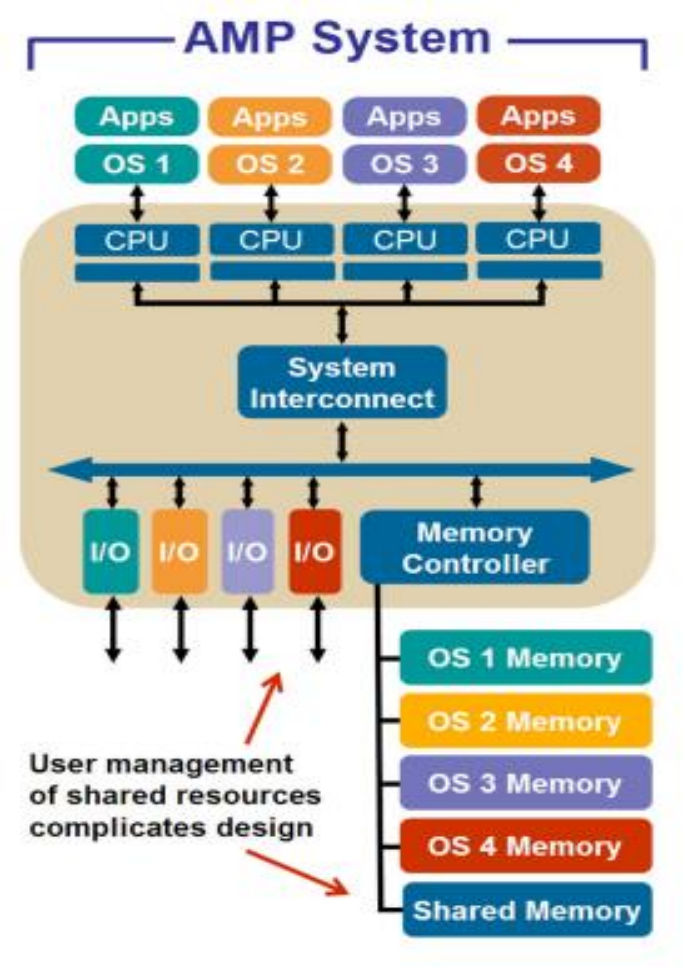
SMP在实时系统潜在的问题

- 无法预测单个进程的执行顺序。
- 无法预测哪个CPU 核执行哪个进程。
- 对于某些进程，可能无法保证实时行为。
- 如果从现有的单处理器设计移植到多处理器，可能会出现同步或互斥问题。
- 业界已开发绑定多处理（BMP）以最大程度地减少此类问题。

实时操作系统的SMP实现方法有一定的改进

非对称多处理器AMP的特点

- 每个CPU内核执行专门的任务，操作与单处理器设计完全相同。
- 系统的行为和可预测性与单处理器设计相似。
- 如果软件针对特定类型的硬件设计，在非对称多处理器上使用异构AMP。
- 在需要混合使用操作系统和/或调度方法的场景，在对称多处理器上使用同构AMP。
- 所有方法都应提高性能为目标。
- 安全是另外一个被考虑的因素。



图片来自： Case Study — Making a Successful Transition to Multi-Core Processors /

RTOS的任务调度与资源共享

周期性调度

重点

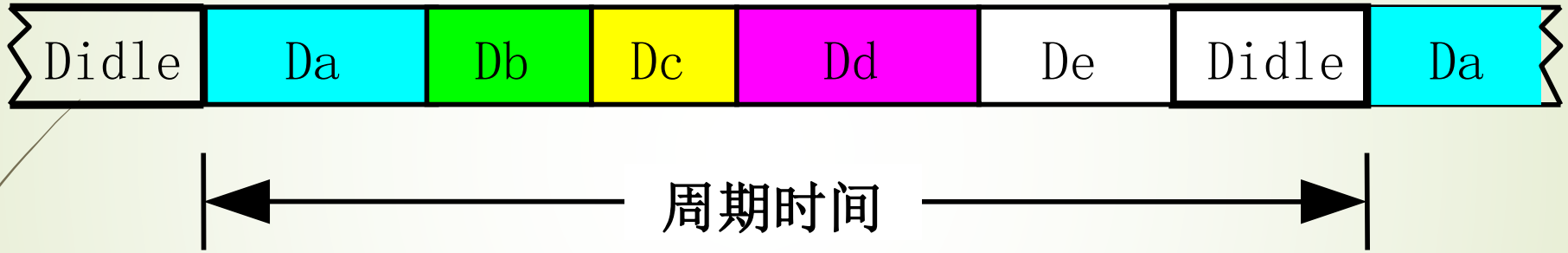
- 每个任务都运行到结束
- 任务没有时间限制
- 先入先出（FIFO）任务执行
- 在分时系统中常见，不在实时嵌入式系统中使用

先到先得（FCFS）

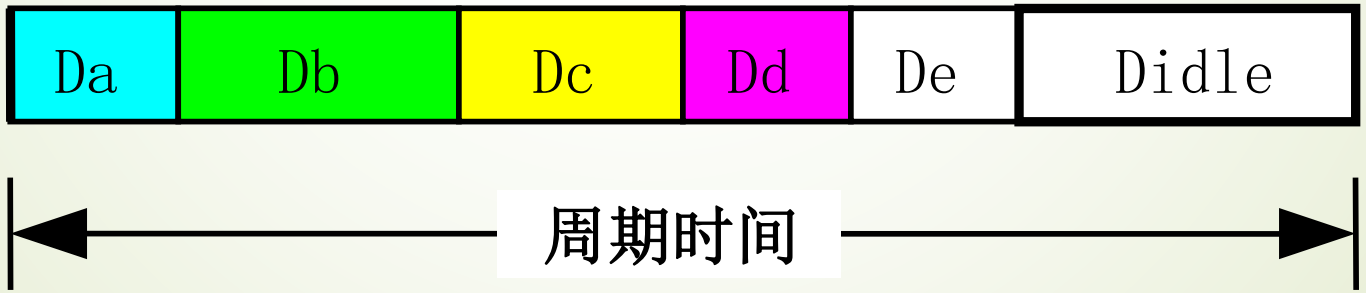
运行到结束

周期循环调度- 引入时间限制

- Didle任务用于填充CPU空闲的时间。
- 以下框架中：
  - 系统运行时，任务会重复执行。
  - 每次任务执行都会运行到结束。

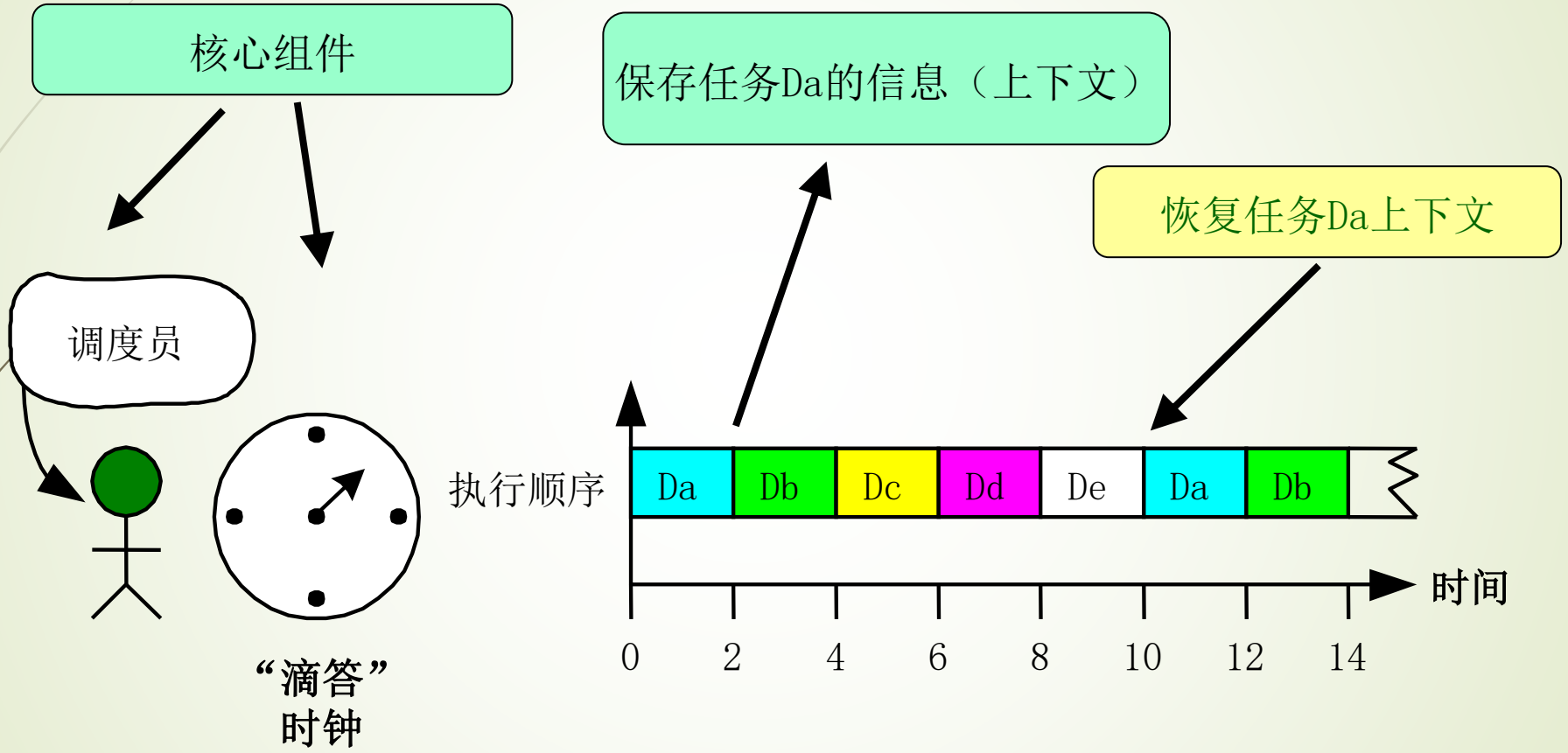


- Didle时长可以调整，以保证固定的周期时间：



- 所有任务必须在周期时间内执行完成。

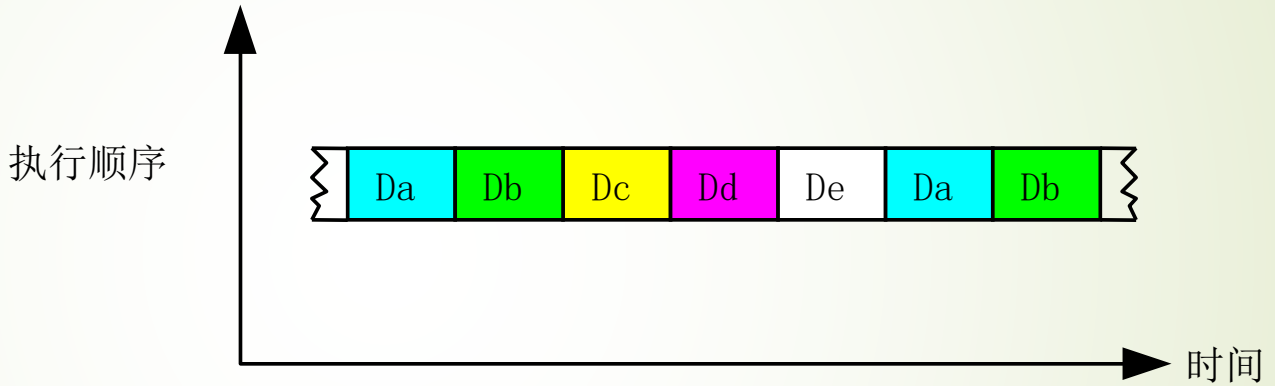
轮询调度一对任务进行时间切片



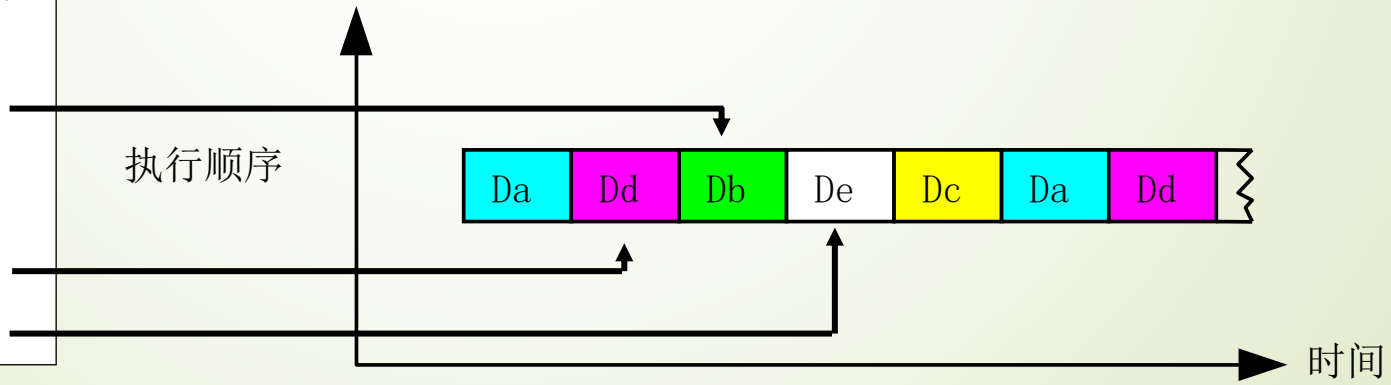
任务的优先级

- 任务可以有优先级。
- 这里优先级用于定义任务的运行时执行顺序，高优先级任务先执行。

任务	优先级
Da	1
Db	2
Dc	3
Dd	4
De	5

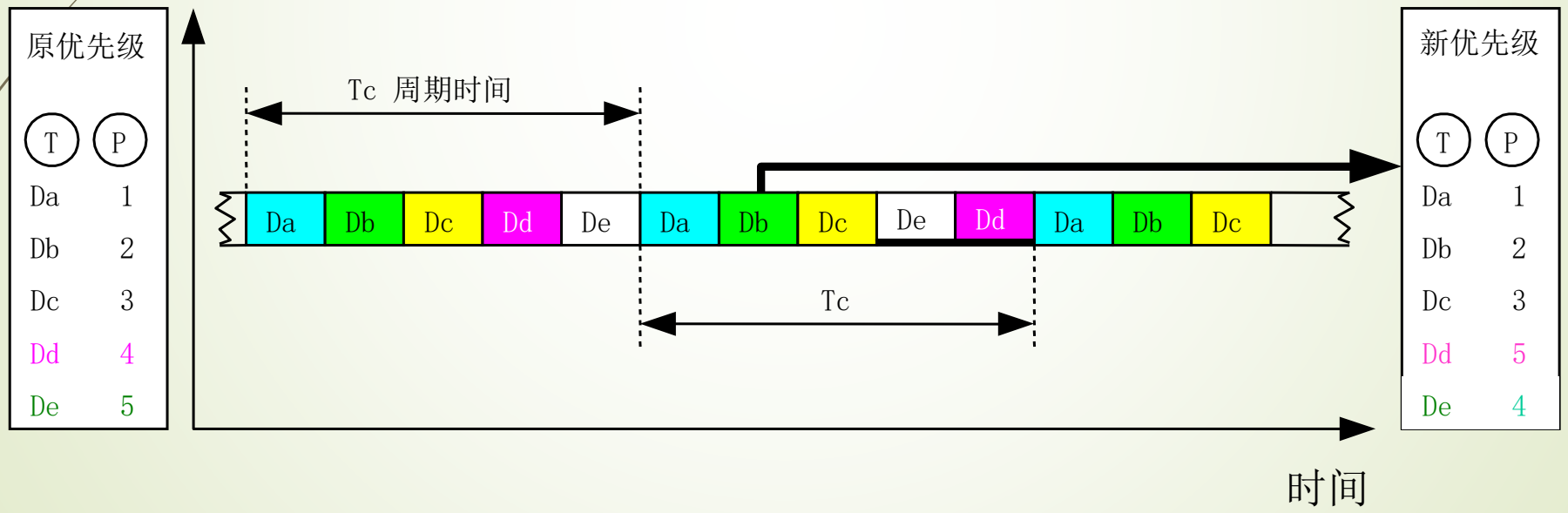


任务	优先级
Da	1
Db	3
Dc	5
Dd	2
De	4



动态优先级改变

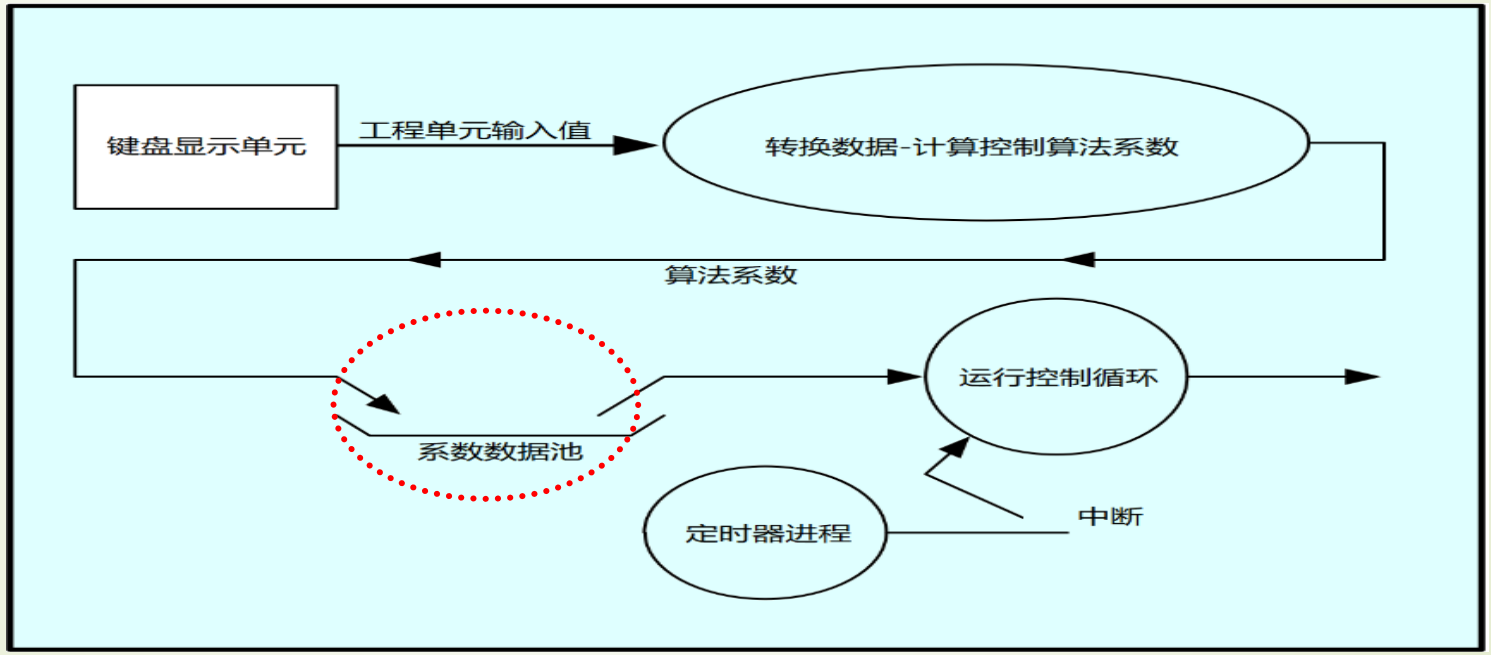
- 有时任务运行时的顺序需要变更，可能的原因有：
  - 任务协作需要
  - 系统模式改变
  - 任务同步需要



共享资源使用中的问题

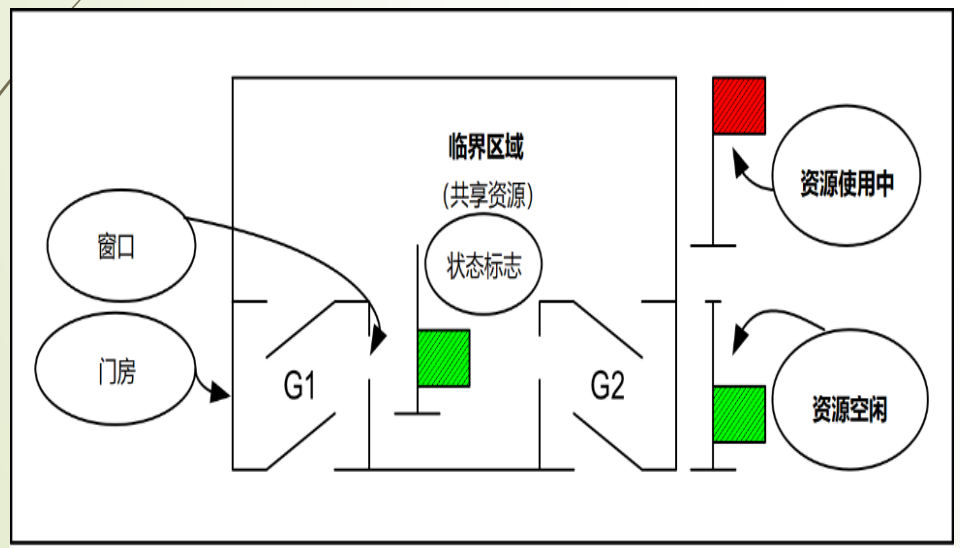
参考图书第3章：使用互斥机制控制资源共享

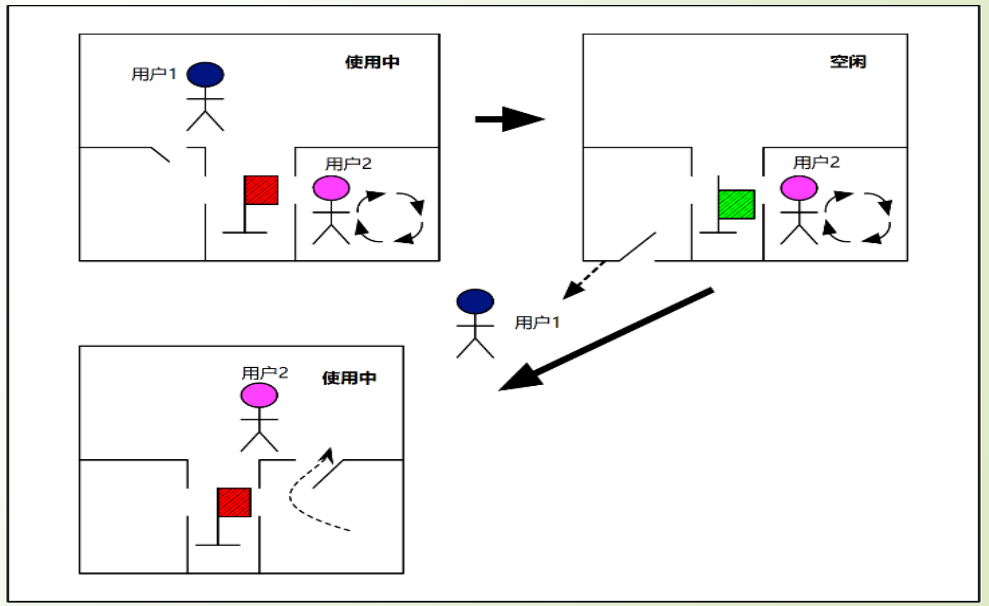
- 在单 CPU 系统中,处理器是一个共享资源。在多个进程之间共享处理器时,处理器的使用由调度程序控制,不存在竞争问题。但系统的其他资源情况并非如此。不同的任务可能需要同时使用同一硬件外设或内存区域。如果不控制这些公共资源的访问,系统中很快就会出现资源争用问题。



使用互斥机制控制资源共享

- 为需要使用资源的每个任务提供一个门房,作为其使用资源的手段。通过标志指示器指示任务是否在房间内(临界区域)。每个任务从门房内只能看到标志升起或降下。
- 用户（任务）发现资源可用,于是它升起标志并进入临界区域,成为共享资源的唯一拥有者，最终用户完成工作离开,它的最后一份工作是降下标志,表示资源被释放。





在多核/多处理器中使用标志时，标志也称为自旋锁

多核的SMP资源共享与实时调度

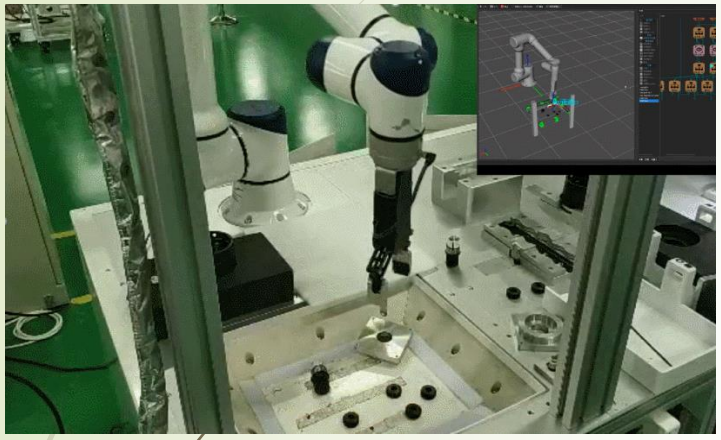
除了解决单核处理器调度和资源共享问题以外，多核处理器架构下,多个核之间解决互相竞争，共享核间资源，及核间存在同步和互斥等问题。

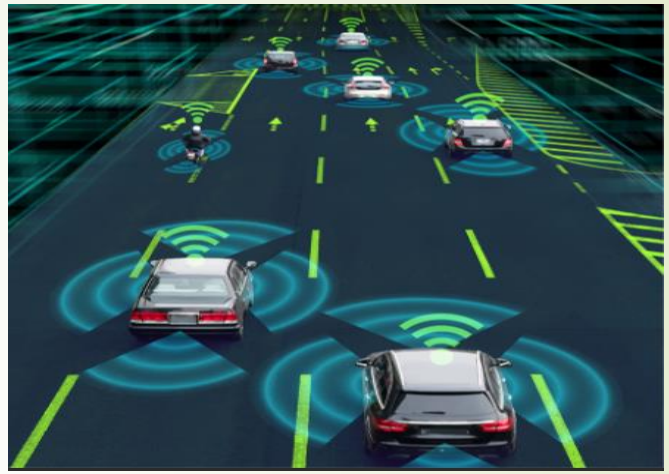
- 临界区保护（自旋锁算法）
  - 自旋锁提供了互斥机制用于保护内核临界区,只有持有自旋锁的处理器核才能执行临界区代码,没有获得自旋锁的处理器核只能进入自旋等待或者休眠状态,直到持有者释放锁。
  - 为了提高效率，自旋锁的算法在不断的改进。
- 全局调度机制（多核调度算法）
  - 任务调度过程分就绪队列操作、调度算法执行和任务上下文切换三个阶段。
  - 各核之间通过竞争来操作任务就绪队列，必须使用锁来确保互斥性，频繁调度和竞争- 时延
  - 调度算法执行会从任务就绪队列中挑选出优先级最高的N个任务分配给N个处理器核。
  - 任务上下文切换主要工作是根据调度算法的执行结果,向需要切换执行任务的处理器发送核间中断,处理器收到中断后上下文进行保存,退出中断处理流程时上下文恢复,完成了任务切换。

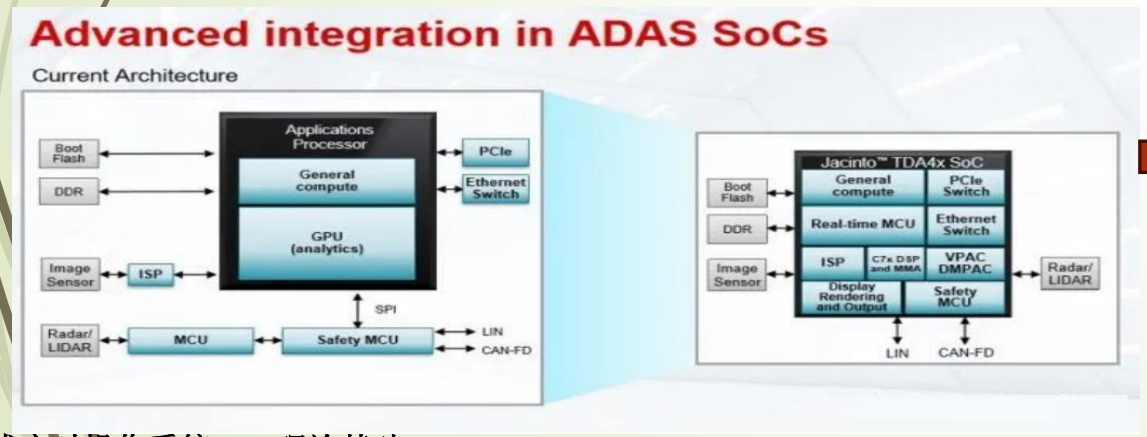
参考:
①赵婉芳 陈莉莉 基于多核处理器的 RTOS 系统分析探究
② rt-thread-v4.1.0 代码 和 K210 BSP

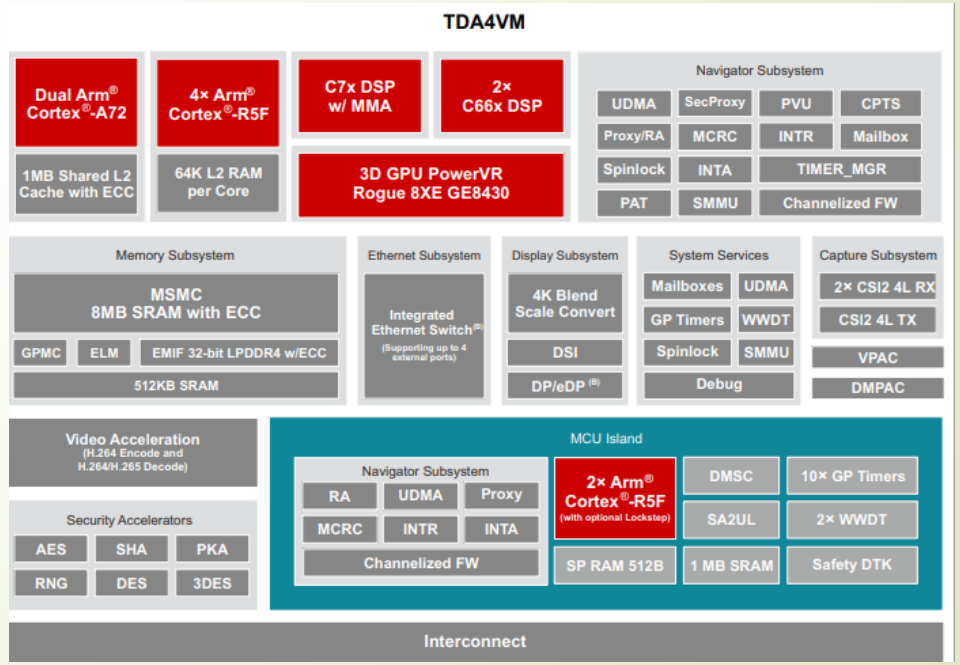
多核处理器实时操作系统应用

多核处理器与RTOS的应用





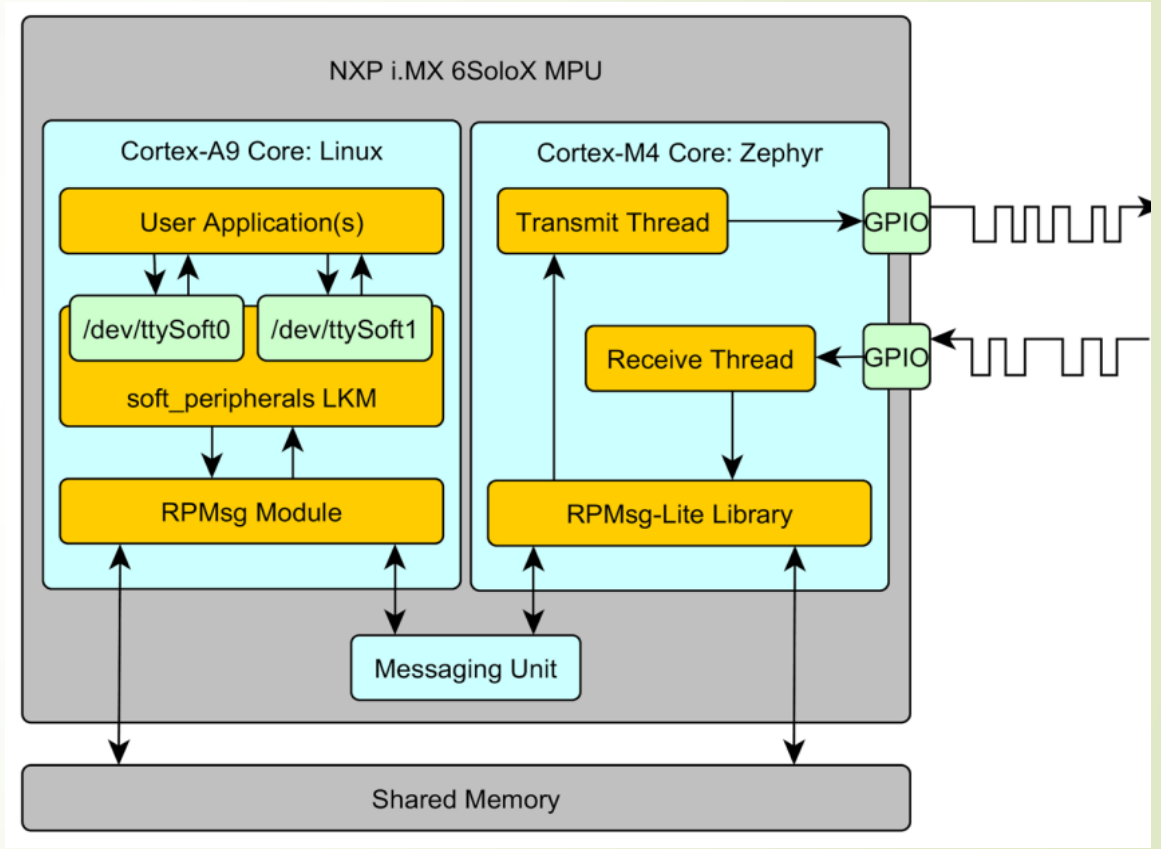




来自：TDA4VM Jacinto™ Processors for ADAS and Autonomous Vehicles Silicon

软件定义外设
-在 Linux 中使用 Zephyr和 RPMsg

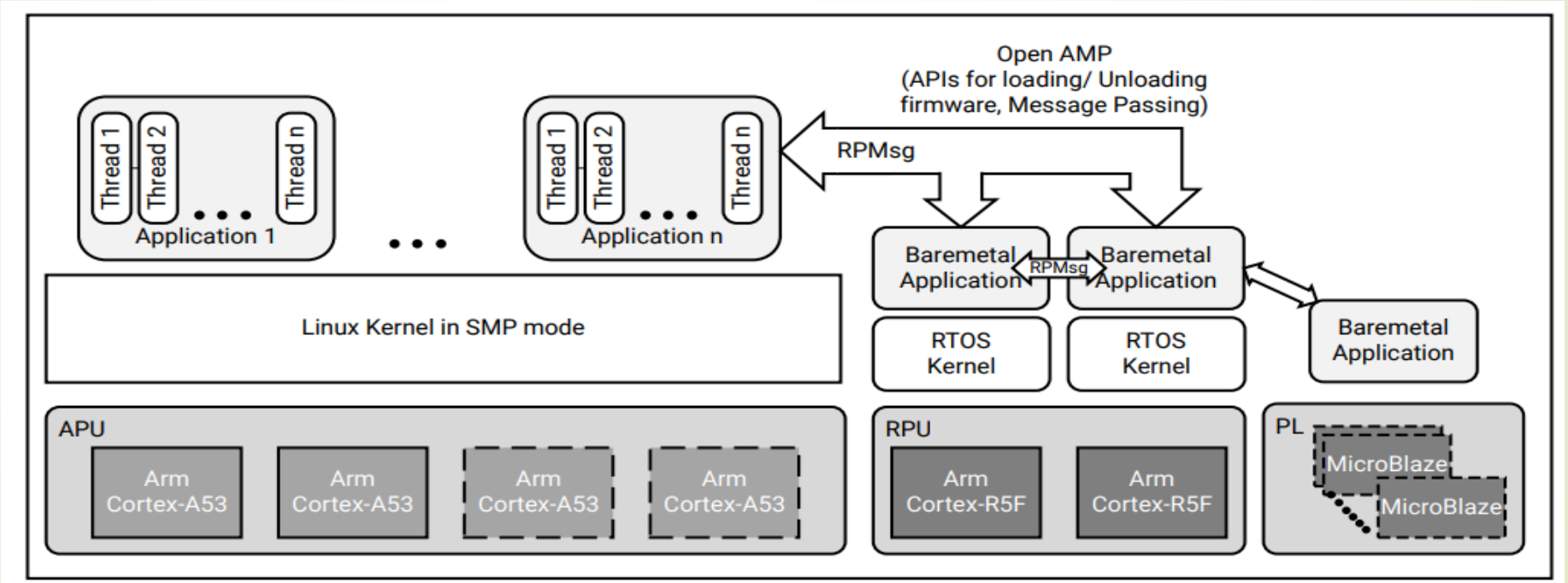
- 在典型的嵌入式 SoC 中，低速外设的数量（如2C、SPI、UART、CAN、LIN、KLINE、DALI、MBUS）有限制。
- i.MX 6SoloX的SoC包含实时协处理器可实现低速异步外设（UART 115200波特率）或中速同步外设（数MHz 的 SPI）。
- 使用多核通信将 多核“外围设备” 映射到到 Linux 设备管理中。
- 运行 Cortex-M4 协处理上的操作系统是 Zephyr RTOS
- 多核之间的通信是基于共享内存和中断机制（RPMsg- 远程处理器消息）
- Linux 、Zephyr 和RPMsg 均为开源软件。



来自：Marek NOVAK 在 EW2019 演讲

使用OpenAMP框架 的SMP 和AMP

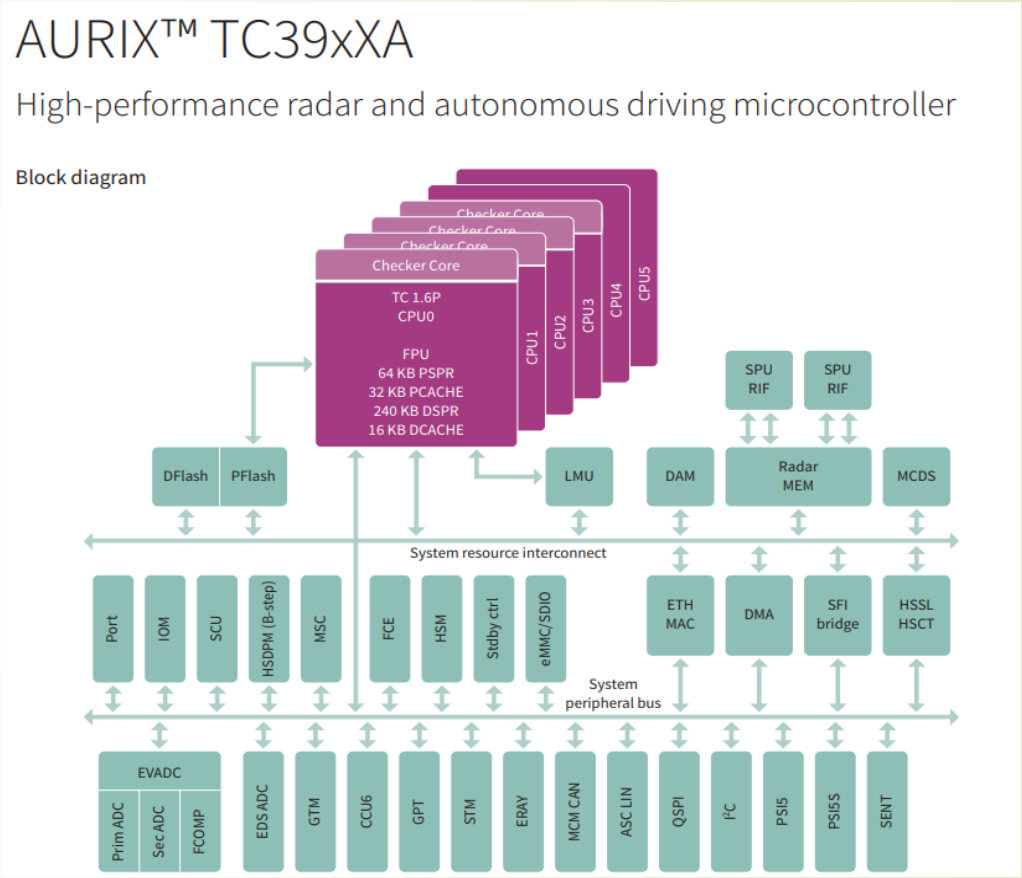
- Xilinx Zynq UltraScale+ MPSoC 架构支持异构多处理器 可以作为针对不同应用的任务的处理器引擎， 提供了AMP和SMP 组合的解决方案。



来自：Zynq UltraScale+ MPSoC Software Developer Guide

基于TriCore FreeRTOS/RT-Thread SMP 汽车电子应用

- 6 个TriCore 运行在 300 MHz 主频。
- 芯片过了ISO 26262 ASIL-D 认证。
- 生态上AUTOSAR 4.2 支持。
- 雷达集群应用
  - LVDS雷达接口
  - 锁步雷达处理器
  - 高带宽雷达SRAM
- 社区做了Free RTOS SMP Infineon 第二代Tricore处理器移植。
- 社区做了RT-Thread 移植（TC397)。



TC397A移植FreeRTOS_tc297 rtos-CSDN博客

SMP: Add task creation with affinity functions by Dazza0 · Pull Request #470 · FreeRTOS/FreeRTOS-Kernel · GitHub

基于Tricore架构的RTThread多核实现-电子工程专辑 (eet-china.com)

实时操作系统SMP 支持情况

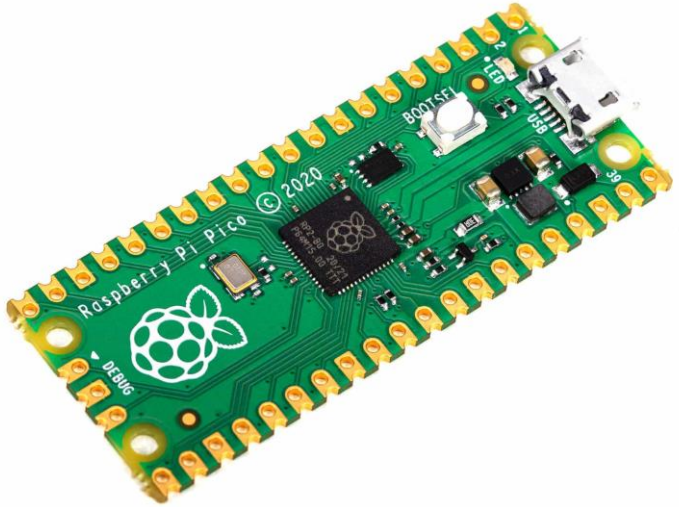
多数RTOS 支持 AMP 处理器架构， SMP 支持越来越多：

- 开源RTOS
  - Zephyr，官方文档宣称支持
  - Nuttx，官方文档宣称支持
  - FreeRTOS 官方文档宣称支持
  - RT-Thread，社区里有讨论，比如 Paspberry-Pico SMP调度移植
- 商业RTOS
  - Thread-X 官方文档宣称支持
  - QNX 多核处理器支持全面，SMP,BMP,AMP
  - VXWORKS 官方宣称支持
  - SylixOS 官方宣称支持
  - RT-Thread 商业版本支持

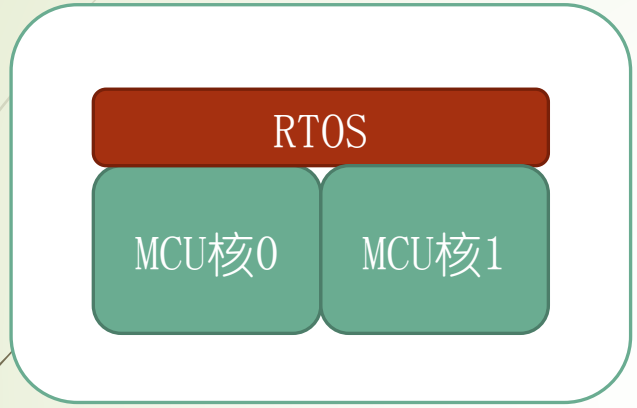
在树莓派Pico上通过FreeRTOS实现对称多处理

树莓派Pico

- RP2040微控制器
- 133MHz双核Arm Cortex M0+
- 264KB SRAM
- 2MB Flash
- GPIO、UART、SPI和I2C支持
- 温度传感器
- WiFi和蓝牙支持（Pico W和WH）

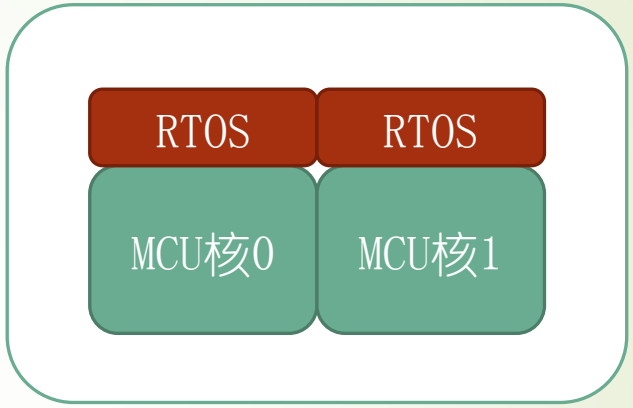


对称和非对称多处理



对称多处理（SMP）：

- 一个RTOS在多个处理器核上运行任务
- 任务负载由多个核分担，动态任务调度
- 处理器核可以共享内存区域



非对称多处理（AMP）：

- 每个处理器核上运行一个RTOS
- 每个核的行为和单核系统类似
- 进一步分为同构和异构AMP

FreeRTOS和SMP支持

- FreeRTOS是一款有20年历史的开源轻量级RTOS，支持40余种体系结构。
- 标准FreeRTOS发行版支持单核处理器，其他一些RTOS已经支持SMP，如VxWorks、Zephyr和PX5。
- FreeRTOS非官方的SMP支持：Espressif的ESP-IDF针对ESP32双核处理器SoC对FreeRTOS进行了定制。
- 2021年宣布的FreeRTOS官方SMP分支支持多个相同体系结构的处理器核；处理器核必须都能够访问同一片内存空间。
- 使用FreeRTOS的对称多处理(SMP)时，一个FreeRTOS实例跨多个处理器核心调度RTOS 任务。
- 和标准发行版相比，SMP分支额外增加8个与处理器亲和性和抢占有关的API以及数个配置宏。

FreeRTOS调度策略（单核）

FreeRTOS使用抢占式调度策略，同等优先级的任务执行时间片轮询调度：

- 任务的初始优先级在编译时确定。
- “抢占式调度”是指调度器始终运行优先级最高且可运行的任务，无论任务何时能够运行。高优先级任务就绪时，调度器会停止当前正在运行的低优先级任务并启动高优先级任务——即使这发生在同一个时间片内。此时，高优先级任务“抢占”了低优先级任务。
- “轮询调度”是指具有相同优先级的任务轮流进入运行状态。
- “时间片”是指调度器会在每个滴答（tick）中断时在同等优先级任务之间进行切换，两次中断之间的时间构成一个时间片。滴答中断是RTOS用来衡量时间的周期性中断。

FreeRTOS调度策略（单核）

抢占式调度和任务饥饿

- 总是运行优先级最高且可运行的任务的后果是，永远不会进入“阻塞”或“挂起”状态的高优先级任务会让所有低优先级任务永久饥饿。
- 解决方法是创建事件驱动型任务：
  - 如果一个高优先级任务正在等待一个事件，该任务进入“阻塞”状态来等待事件。高优先级任务处于“阻塞”状态时，低优先级任务会运行。
  - 事件“发生”时，可以使用众多FreeRTOS任务间通信和同步原语之一将事件发送给任务。接收到事件后，高优先级任务会自动解除“阻塞”状态并抢占执行。

配置调度策略

- 通过以下FreeRTOSConfig.h的设置更改默认调度行为：
  - configUSE_PREEMPTION：如果设置为0，则关闭抢占，只有当运行状态的任务进入“阻塞”或“挂起”状态，运行状态任务调用 taskYIELD()，或中断服务程序（ISR）手动请求上下文切换时，才会发生上下文切换。
  - configUSE_TIME_SLICING：如果设置为0，则关闭时间切片，调度器不会在每个滴答中断时在同等优先级的任务之间切换。

SMP和任务调度

- SMP中的调度方法和单核一致，但是同时可以有等同于处理器核数量的任务处于运行状态。
  - 这意味着，可能同时有较高和较低优先级的任务在不同核上运行，即低优先级的任务不一定要等待所有高优先级任务结束才能运行。
  - 例如，有一个高优先级任务和两个中等优先级任务处于“就绪”状态，SMP调度器需要选择两个任务，每个核心对应一个任务。首先，选择较高优先级任务在第一个核心上运行。这样就剩下了两个中等优先级的任务作为可运行的任务，因此会在第二个核心上轮询运行。结果是高优先级和中等优先级的任务同时运行。
- 同优先级的任务由于核亲和度不同，不一定会严格按序轮询执行。
  - 在所有核上都可以运行的任务依然能够按序轮询。

FreeRTOS SMP配置（FreeRTOSConfig.h）

宏	用途
configNUM_CORES*	设置处理器核的数目。 树莓派Pico：设置为2
configTICK_CORE	配置在哪个处理器核上更新滴答，默认为0。
configUSE_CORE_AFFINITY	0 - 不允许任务设置处理器核亲和性。 1 - 允许任务设置处理器核亲和性。
configRUN_MULTIPLE_PRIORITIES	0 - 只有同样优先级的任务能在不同核上同时运行；只要任一核上有一个高优先级任务，低优先级任务就不能运行。 1 - 不同优先级的任务能在不同核上同时运行。
configUSE_TASK_PREEMPTION_DISABLE	0 - 不允许任务配置自身不被抢占。 1 - 允许任务配置自身不被抢占。 该值为1时configRUN_MULTIPLE_PRIORITIES必须为1。

*SMP分支并入主分支后被更名为configNUMBER_OF_CORES。

FreeRTOS SMP API(1)

API	作用	所需配置
vTaskCoreAffinitySet	设置任务和处理器核的亲和性位掩码，即在哪些核上可以运行任务。	configUSE_CORE_AFFINITY = 1
vTaskCoreAffinityGet	获取任务的处理器核亲和性位掩码。	configUSE_CORE_AFFINITY = 1
vTaskPreemptionDisable	在任务中禁用抢占。	configUSE_TASK_PREEMPTION_DISABLE = 1
vTaskPreemptionEnable	在任务中启用抢占。	configUSE_TASK_PREEMPTION_DISABLE = 1

FreeRTOS SMP API(2)

API	作用	所需配置
xTaskCreateAffinitySet	基于xTaskCreate()，创建一个任务，同时设置任务和处理器核的亲和性位掩码。	configUSE_CORE_AFFINITY = 1
xTaskCreateStaticAffinitySet	基于xTaskCreateStatic()，以静态内存分配方式创建一个任务，同时设置任务和处理器核的亲和性位掩码。	configUSE_CORE_AFFINITY = 1
xTaskCreateRestrictedAffinitySet	基于xTaskCreateRestricted()，创建一个MPU保护任务，同时设置任务和处理器核的亲和性位掩码。	configUSE_CORE_AFFINITY = 1
xTaskCreateRestrictedStaticAffinitySet	基于xTaskCreateRestrictedStatic()，以静态内存分配方式创建一个MPU保护任务，同时设置任务和处理器核的亲和性位掩码。	configUSE_CORE_AFFINITY = 1

ESP-IDF SMP和官方SMP支持的对比

- ESP-IDF（Espressif物联网开发框架）中对FreeRTOS内核进行了修改，添加了SMP支持和相关API。
  - 2016年的ESP-IDF 0.9发布版就已经包含了SMP支持。
  - 通过调用API xTaskCreatePinnedToCore()和xTaskCreateStaticPinnedToCore()，在创建任务的同时，可以指定任务在核0、1，或者两个核上运行（tskNO_AFFINITY）。
  - 由于是针对ESP32定制：不支持多于双核的处理器，缺少与MPU保护任务相关的API。
  - 缺少与抢占相关的配置和API。
- 默认的抢占调度策略被稍作修改：
  - 只有任务亲和的核才能执行任务，不亲和的任务会被跳过。这可能会导致一个核跳过更高优先级，先运行更低优先级的任务。
  - 同优先级轮询时由于亲和性可能会导致先就绪的任务后运行。
  - 这和官方SMP支持类似。
- 核0负责更新滴答，即处理滴答中断。
  - 无法配置，与官方SMP支持不同。

将现有FreeRTOS应用从单核转为SMP——注意事项

- 应用不能假设只有高优先级任务完成之后低优先级任务才能运行。
  - 当两个不同优先级的任务共享资源时，不能只依赖优先级保证资源的使用顺序，最佳方法是使用同步机制（信号量等）进行互斥。
  - 不改变任务自身代码的情况下，可以通过vTaskCoreAffinitySet() API将共享资源、优先级不同的任务绑定到一颗处理器核上，保证这些任务不会同时运行。
  - 也可以通过将configRUN_MULTIPLE_PRIORITIES设置为0避免高、低优先级的任务同时运行，但是这样会造成处理器资源的浪费。例如，只有一个高优先级任务，但是有多个低优先级任务时，只有一个核会运行高优先级任务。
- 应用不能假设同周期、同优先级的任务会按照固定顺序轮询运行。
- 中断例程（ISR）会和其他处理器核上的任务同时运行。ISR和任务共享资源时需要互斥保护。
  - FreeRTOS提供一套宏帮助ISR或任务进入一个禁用中断的关键区域，此时任务轮询调度和抢占也不会发生。在这一区域内可以访问ISR和任务共享的资源。
  - 关键区域不可用于在不同核上运行的任务之间的互斥。
  - ISR调用taskENTER_CRITICAL_FROM_ISR()和taskEXIT_CRITICAL_FROM_ISR()宏进入/离开关键区域；任务调用taskENTER_CRITICAL()和taskEXIT_CRITICAL()宏进入/离开关键区域。

样例代码：构建和运行（1）

- 以Ubuntu系统为例：
- 工具链和其他开源工具：
  $ sudo apt install git cmake gcc-arm-none-eabi build-essential minicom
- 下载FreeRTOS和Pico SDK（需要环境能够访问GitHub）：
  $ mkdir freertos-pico
  $ cd freertos-pico
  $ git clone https://github.com/RaspberryPi/pico-sdk --recurse-submodules
  $ git clone -b smp https://github.com/FreeRTOS/FreeRTOS-Kernel --recurse-submodules
  $ export PICO_SDK_PATH=$PWD/pico-sdk
  $ export FREERTOS_KERNEL_PATH=$PWD/FreeRTOS-Kernel
- 注意：检查最后两个环境变量必须指向freertos-pico下的正确目录，而且每次启动命令行会话都要重新设定。

样例代码：构建和运行（2）

- 复制smpTasks和taskPreemption目录到freertos-pico下。
- 以smpTasks为例，构建代码：
  $ cd smpTasks
  $ cmake .
  $ make
- 构建成功时会生成smpTasks.uf2文件。
- 在按下Pico BOOTSEL白色按钮的同时将Pico连接到系统上。Pico会被识别为一个U盘，一般路径为“/media/<用户名>/RPI-RP2”。
- 用文件管理器拖拽或者cp命令将uf2文件复制到U盘中。Pico会重启开始执行程序。
  $ cp smpTasks.uf2 /media/<用户名>/RPI-RP2/
- 用minicom连接串口读取输出：
  $ sudo minicom -b 115200 -o -D /dev/ttyACM0

代码样例1：多任务和处理器核亲和性

- 创建A、B、C、D四个任务，优先级相同。
- A、B通过两种不同API设置处理器亲和性，只能分别运行在核0/1上。C、D不设置亲和性。
- 运行中A、B只会分别运行在核0和核1上，C、D运行的核不定。
- 样例输出：

```
Task A core 0 tick 5700 mask 1
Task D core 1 tick 5701 mask -1
Task B core 1 tick 5701 mask 2
Task C core 1 tick 5800 mask -1
Task A core 0 tick 5800 mask 1
Task D core 0 tick 5801 mask -1
Task B core 1 tick 5801 mask 2
…
Task A core 0 tick 6200 mask 1
Task D core 0 tick 6201 mask -1
Task B core 1 tick 6201 mask 2
Task C core 1 tick 6300 mask -1
Task A core 0 tick 6300 mask 1
Task D core 1 tick 6301 mask -1
```

代码样例2：禁用抢占

- 创建A、B、C三个任务，B和C的优先级较A高。任务A只能运行在核0上，B、C不设置亲和性。
- 任务A持续执行，每3秒周期性禁用或允许抢占。任务B、C每1秒会进入可调度状态。
- A禁用抢占时，B、C会在核1上依次运行；A允许抢占时，B、C同时占用两个核运行，即A被抢占。
- 样例输出：

```
Task A priority 1 core 0 mask 1 tick 15005 - preemptable
Task B priority 2 core 1 mask -1 tick 16000
Task C priority 2 core 0 mask -1 tick 16001
Task B priority 2 core 1 mask -1 tick 17000
Task C priority 2 core 0 mask -1 tick 17001
Task B priority 2 core 1 mask -1 tick 18000
Task C priority 2 core 0 mask -1 tick 18001
Task A priority 1 core 0 mask 1 tick 18006 - no preempt
Task B priority 2 core 1 mask -1 tick 19000
Task C priority 2 core 1 mask -1 tick 19001
Task B priority 2 core 1 mask -1 tick 20000
Task C priority 2 core 1 mask -1 tick 20001
Task B priority 2 core 1 mask -1 tick 21000
Task C priority 2 core 1 mask -1 tick 21001
```

FreeRTOS SMP分支合并

- FreeRTOS单独的SMP代码分支2023年10月并入了FreeRTOS主分支，即未来新的标准发布版会包含SMP支持。
  - SMP分支将不再更新。
  - 主分支的Pico板FreeRTOS移植支持也更新为默认使用SMP。
- 合并过程中configNUM_CORES宏被更名为configNUMBER_OF_CORES。
  - 宏的作用见“FreeRTOS SMP配置”页。
- 本课程的代码样例需要将FreeRTOSConfig.h中的configNUM_CORES宏更名，即可和默认FreeRTOS分支的代码进行编译。
  - 下载默认FreeRTOS代码的命令：git clone https://github.com/FreeRTOS/FreeRTOS-Kernel --recurse-submodules

参考链接——FreeRTOS SMP和树莓派Pico

- FreeRTOS SMP综述：https://www.freertos.org/symmetric-multiprocessing-introduction.html
- FreeRTOS调度：https://www.freertos.org/single-core-amp-smp-rtos-scheduling.html
- FreeRTOS Pi Pico SMP上手：https://www.freertos.org/smp-demos-for-the-raspberry-pi-pico-board.html
- Pico SDK：https://github.com/raspberrypi/pico-sdk
- FreeRTOS SMP的样例和API、配置的解释：https://github.com/FreeRTOS/FreeRTOS-SMP-Demos/blob/main/SMP.md
- ESP-IDF的SMP支持：https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos_idf.html
- AWS Daniel Gross在SCALE 20x会议上的演讲（2023年3月）：https://www.socallinuxexpo.org/sites/default/files/presentations/SCaLE%202023%20-%20FreeRTOS%2BPico.pdf

课件和代码将上传到何小庆老师图书页面

谢谢

- www.hexiaoqing.net
- Allan@esbf.org

何小庆老师图书资料下载

欢迎关注何小庆老师团队嵌入式图书和培训课程（网易云课堂）