极客时间 训练营

系统架构设计师

第16章 嵌入式架构设计理论与实践

授课：王建平

目录

目录

嵌入式系统发展历程

发展阶段	硬件	软件	主要特点
SCM **单片微型计算机**	单片机	无操作系统 汇编语言	结构和功能相对单一 处理效率低 存储容量十分有限 几乎没有用户接口
MCU **微控制器**	单片机 嵌入式微处理器 外围电路 接口电路	以简单操作系统为核心	微处理器微控制器种类繁多 通用性比较弱 系统开销小，处理效率高 智能化控制能力突出
SoC **片上系统**	嵌入式微处理器	嵌入式操作系统	嵌入式系统兼容性好 操作系统内核小 处理效率高
以 Internet 为基础的嵌入式系统	嵌入式微处理器	嵌入式操作系统	微处理器集成网络接口 应用域网络环境中
智能化、云技术推动下的嵌入式系统	微型传感器 智能服务设备		低能耗，高速度，高集成，高可信，适应环境广

嵌入式系统硬件组成结构

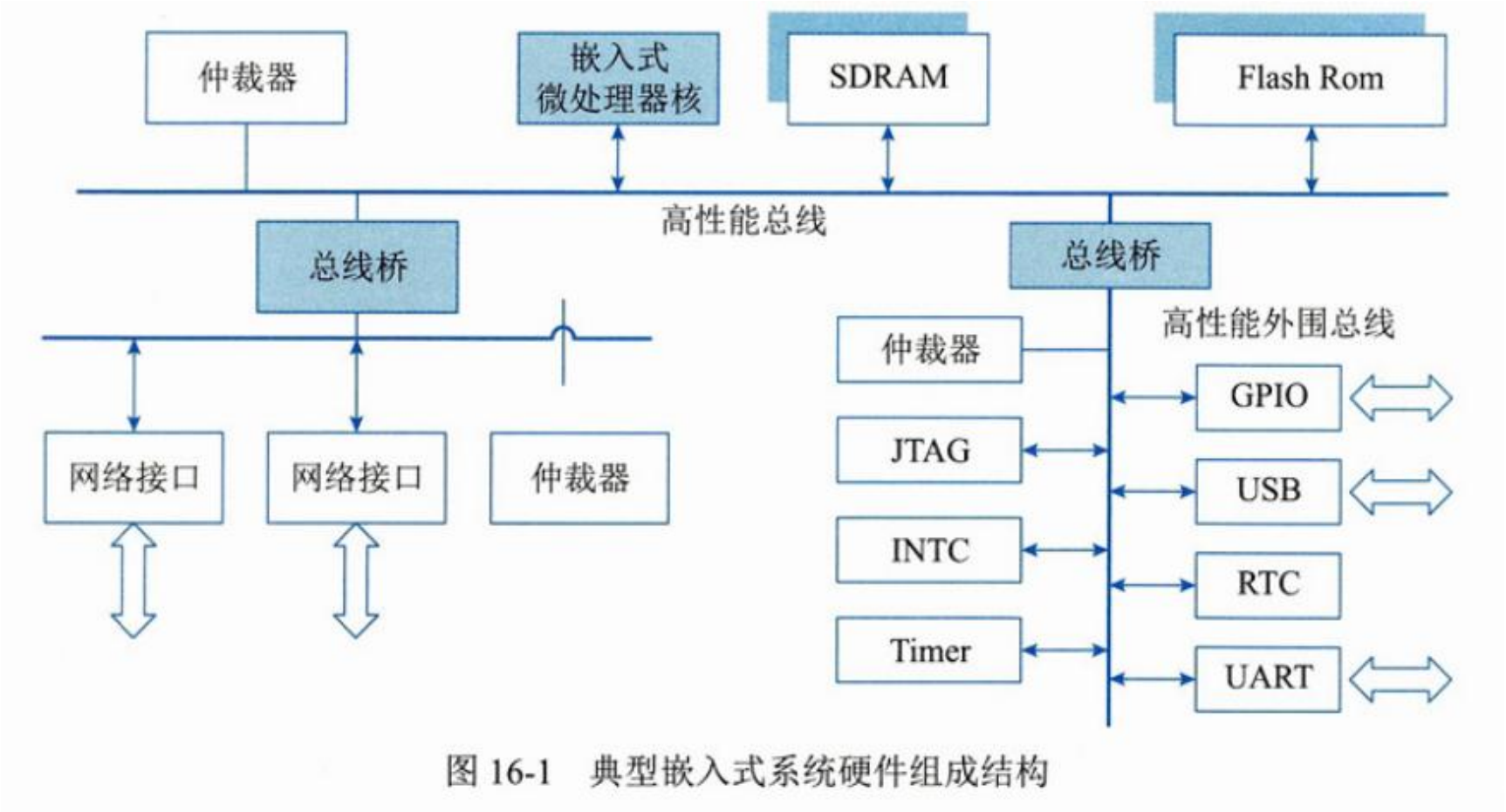
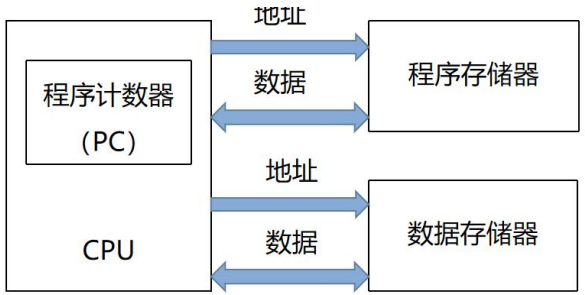


图 16-1　典型嵌入式系统硬件组成结构

嵌入式系统硬件组成结构

1.传统嵌入式系统。主要硬件包括：
- 微处理器：微控制器 MCU，微处理器 MPU
- 存储器：RAM，ROM
- 总线：内总线，外总线
- 定时器/计数器 Timer
- 看门狗 WatchDog
- I/O 接口：串口，网络，USB，JTAG
- 外部设备：UART，LED



2.嵌入式处理器的分类。嵌入式系统可以分为：（★）

（1）微处理器（Micro Processor Unit, MPU)。特点是体积小，重量轻，成本低，可靠性高，但技术保密性差。

（2）微控制器（Micro Control Unit, MCU)。特点是单片化，体积小，功耗低，成本低，可靠性更高。

（3）信号处理器（Digital Signal Processor, DSP)特点是系统结构和指令采用特殊设计，通常采用哈佛结构，编译效率高，指令执行速度也高。

（4）图形处理器（Graphics Processing Unit, GPU)专注于浮点运算，弥补了 CPU 运算速度不足。

（5）片上系统（System on Chip, SoC)，采用了片内再编程技术,可使片上系统内硬件的功能可以像软件一样通过编程来配置,从而可以实时地进行灵活而方便的修改和开发。

嵌入式系统硬件组成结构

3.存储器。存储器的就是一种存储程序和数据用的时序逻辑电路。存储器具有如下分类：（★）
（1）随机存取存储器（ Random Access Memory，RAM）。它的特点是一旦系统断电，存放在里面的所有数据和程序都会自动清空掉，并且再也无法恢复。
（2）只读存储器（Read Only Memory，ROM）。ROM 在元件正常工作的情况下，其中的代码数据将永久保存，并且不能够进行修改。
4.总线。总线是功能部件间传输信息的公共通信干线。总线的拓扑结构有星形、树状、环形、总线型和交叉开关型等五种。总线的类型可以按照计算机所传输的信息种类、按连接部件进行划分。
（1）按照计算机所传输的信息种类可以分为：（★）
数据总线：用于处理器与 RAM 间传输待处理和待存储的数据。
地址总线：用于传输 RAM 中存储数据的地址。
控制总线：用于传输处理器控制单元信号到周边设备。
（2）按连接部件分类（★）
✓ 片内总线：内部总线，连接 ALU，寄存器，指令部件等芯片内部元件。
✓ 系统总线：内部总线，又称板级总线，连接微控制器/处理器，主存，I/O 接口。
✓ 局部总线：内部总线，连接少量组件用于交换数据。
✓ 通信总线：外部总线，又称外设总线，连接外部设备或外部系统。
5.看门狗。看门狗为嵌入式系统提供必须的系统恢复能力，在系统发生软件问题和程序跑飞时重新启动系统。（★）

嵌入式系统软件架构概述

◆通用开放架构：GOA（Generic Open Architecture）架构的主要特点如下：（★）
(1)可移植性。各种计算机应用系统可在具有开放结构特性的各种计算机系统间进行移植，不论这些计算机是否同种型号、同种机型。
(2)可互操作性。如计算机网络中的各结点机都具有开放结构的特性，则该网上各结点机间可相互操作和资源共享。
(3)可剪裁性。如某个计算机系统是具有开放结构特性的，则在该系统的低档机上运行的应用系统应能在高档机上运行，原在高档机上运行的应用系统经剪裁后也可在低档机上运行。
(4)易获得性。在具有开放结构特性的机器上所运行的软件环境易于从多方获得，不受某个来源所控制。

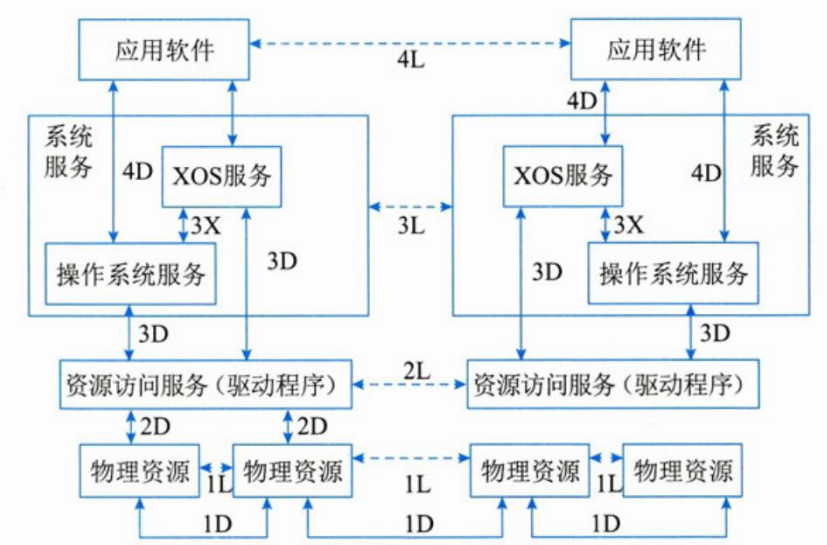


图 16-4　SAE AS4893 定义的一种 GOA 架构

图 16-3　简单的嵌入式软件架构

典型真题

嵌入式系统原在高档机上运行的应用系统也可在低档机上运行体现了嵌入式系统架构（）特点。
A.可裁剪性　B.可移植性　C.复用性　D.互操作性
参考答案：A

目录

两种典型的嵌入式系统架构模式

◆嵌入式系统的典型架构可概括为两种模式，即层次化模式架构和递归模式架构。（★）
1.层次化模式架构：位于高层的抽象概念与低层的更加具体的概念之间存在着依赖关系。
层次化模式架构主要设计思想是：
(1)当一个系统存在高层次的抽象，这些抽象的表现形式是一个个的抽象概念，而这些抽象概念需要具体的低层概念进行实现时，就可采用层次化模式。
(2)分层模式结构只包含了一个主要的元素(域包)和它的接口，以及用来说明模式结构的约束条件。
(3)层次化模式可以分为两种：封闭型和开放性。封闭型的特征是：一层中的对象只能调用同一层或下一个底层的对象提供的方法。而开放型一层中的对象可以调用同一层或低于该层的任意一层的对象提供的方法。

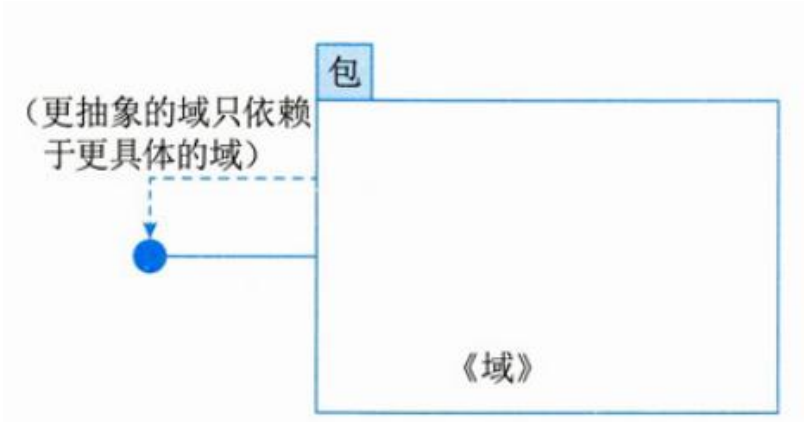


图 16-5　层次化模式架构的示意图

两种典型的嵌入式系统架构模式

2.递归模式架构。递归模式解决的问题是：需要将一个非常复杂的系统进行分解，并且还要确保分解过程是可扩展的，即只要有必要，该分解过程就可以持续下去。（★）

◆在创建这种模式的实例时，通常使用两种相反的工作流程。

● 自顶向下： 自顶向下的工作流从系统层级开始并标识结构对象，这些对象提供实现协作的服务。在实时系统和嵌入式系统中，大多数情况下是基于某个标准方法，将系统分成一个个子系统。当开发人员逐步降低抽象层级，向下推进时，容易确保开发者的工作没有偏离用例中所规定的需求。

● 自底向上： 自底向上专注于域的构造——首先确定域中的关键类和关系。这种方法之所以可行是因为：开发者以往有丰富的开发经验，并能将其他领域所获得的知识映射到当前开发所在的域中。通过这种方法，最终开发者会到达子系统级的抽象。

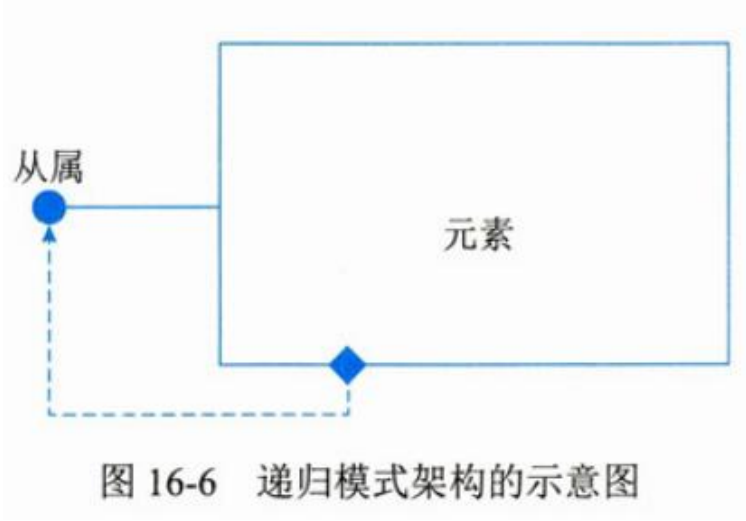


图 16-6　递归模式架构的示意图

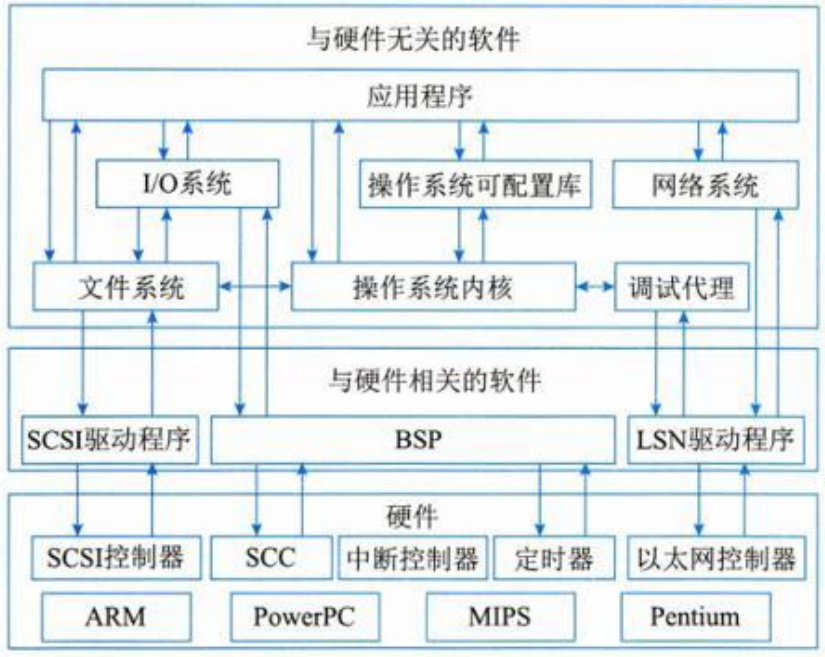
嵌入式操作系统

◆嵌入式操作系统的定义及特点。嵌入式操作系统（Embedded Operating System，EOS）是指用于嵌入式系统的操作系统。与通用的操作系统相比，嵌入式操作系统具有：可剪裁性，可移植性，强实时性，强紧凑性，高质量代码，强定制性，标准接口，强稳定性，弱交互性，强确定性，操作简捷、方便，较强的硬件适应性，可固化性的特点。（★）

- 可移植性：操作系统通常可运行在不同体系结构的处理器和开发板上；
- 强实时性：嵌入式操作系统实时性通常较强，可用于各种设备的控制；
- 强紧凑性：由于嵌入式系统的资源受限的特点，嵌入式操作系统代码需要紧凑、精炼，不应存在无用代码；
- 强定制性：嵌入式操作系统可根据目标系统的不同需求，进行专业化定制；
- 标准接口：嵌入式操作系统可提供设备统一的驱动接口；
- 强稳定性、弱交互性：嵌入式系统一旦运行就不需要用户过多干预，这就要负责管理的操作系统具有较强的稳定性。EOS的用户接口一般不提供操作命令，它是通过系统的调用命令向用户程序提供服务的；
- 强确定性：EOS对任务调度和资源管理应能够确保其在规定的时间、规定的容量内不发生任务超时和资源枯竭；
- 可固化性：在嵌入式系统中，嵌入式操作系统和应用软件通常是被固化在计算机系统的ROM中，系统运行时调入内存运行。

嵌入式操作系统

◆**嵌入式系统的架构**。嵌入式操作系统分为面向控制、通信领域以及面向消费电子产品两类。
从嵌入式操作系统体系架构看，主要存在4种结构：整体结构、层次结构、客户/服务器结构和面向对象结构。
整体结构的架构图，又称为模块结构或无序结构，基于结构化程序设计的一种软件设计方法。



与处理器硬件相关的驱动称为结构支持包（ASP）
与处理器外围芯片相关的驱动称之为板级支持包（BSP）

嵌入式操作系统

◆嵌入式操作系统的基本功能

（1）操作系统内核架构---操作系统核心部分。（★）

① 宏内核。用于管理用户程序和硬件间的系统资源，在宏内核中用户服务和内核服务在同一空间中实现，代码耦合度非常高，内核的功能组件代码可以互相调用。

② 微内核。微内核管理所有系统资源，在微内核中用户服务和内核服务在不同空间中实现，系统结构清晰，代码量少。

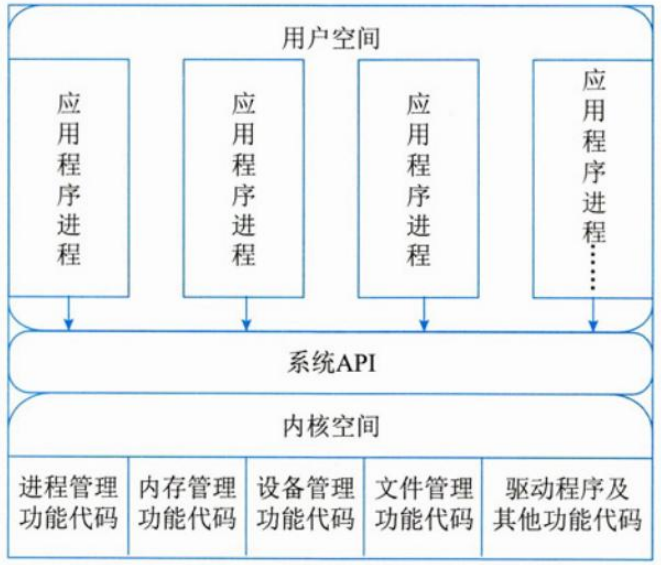


图 16-8　宏内核嵌入式操作系统结构

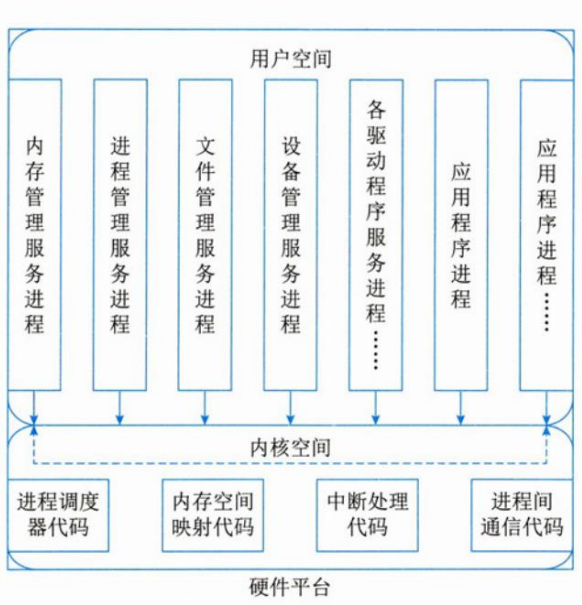


图 16-9　微内核嵌入式操作系统结构

嵌入式操作系统

◆嵌入式操作系统的基本功能

（2）任务管理。任务是嵌入式操作系统调度最小单位为任务，类似于计算机操作系统中进程的概念。任务有三种工作状态：（★）

✓ 执行状态：任务获得处理机，程序在处理机中执行。

✓ 就绪状态：任务已获得处理机以外资源，待获得处理机即可执行。

✓ 阻塞状态：执行状态任务因等待事件发生无法执行而放弃处理机。

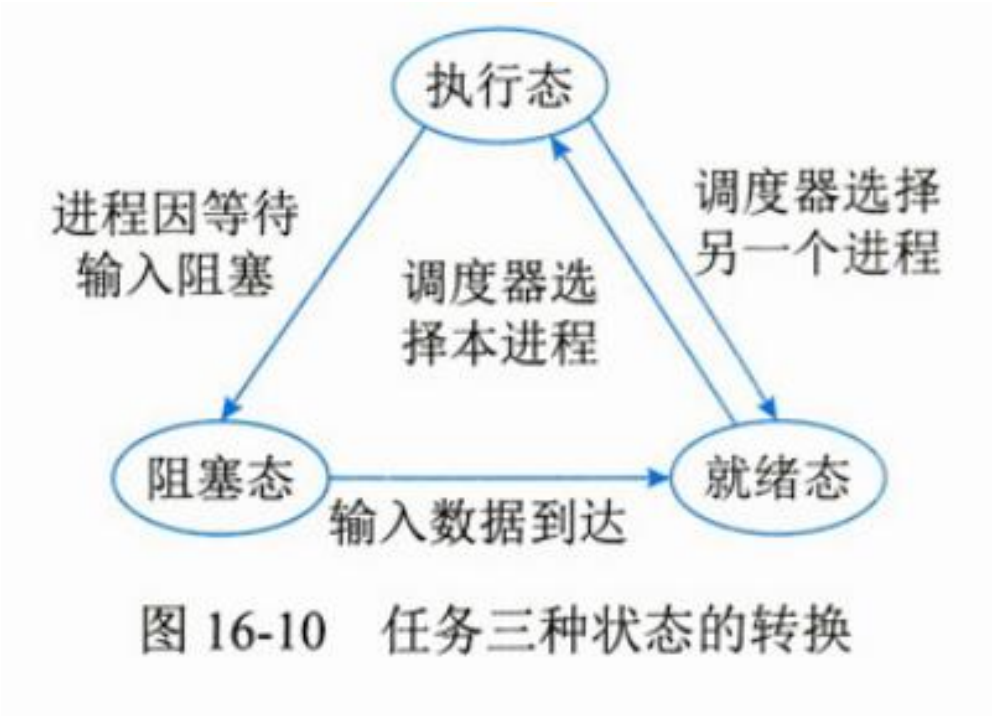


图 16-10　任务三种状态的转换

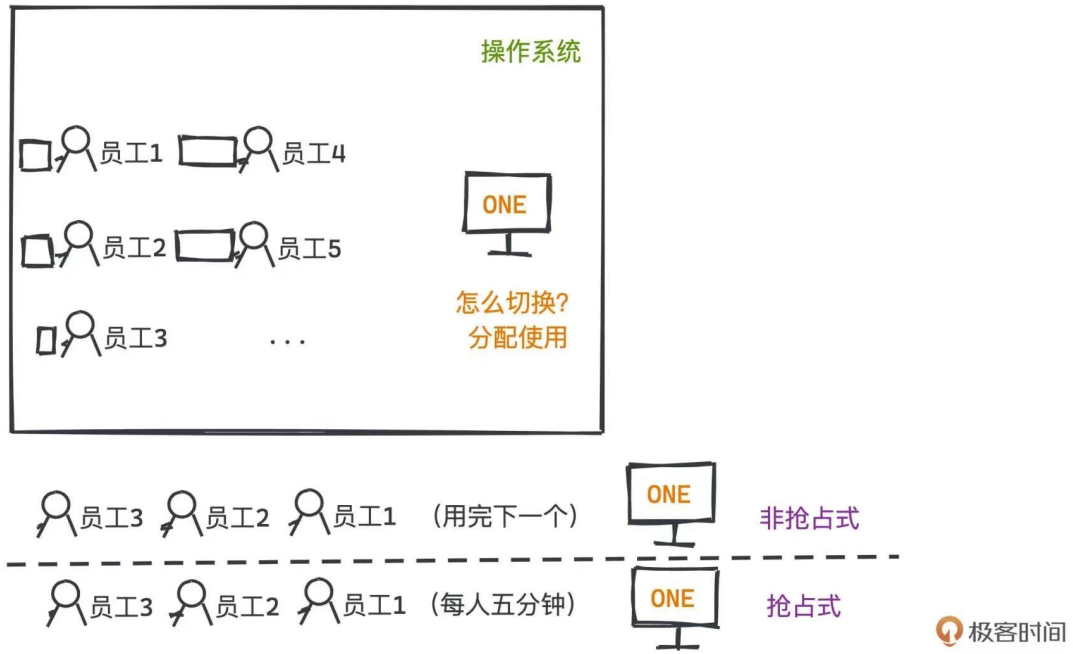
嵌入式操作系统

◆在嵌入式强实时系统中，操作系统必须支持执行一组并发实时任务，使所有时间关键任务满足其指定的截止时限(Deadline)。在实时系统的任务调度中，存在大量的实时调度方法，大致可以概述为主要三种划分，即离线(Off-Line)和在线(On-Line)调度、抢占(Preemptive)和非抢占(Non-Preemptive)调度、静态(Static)和动态(Dynamic)调度等。（★★）

离线	运行过程中使用的调度信息在系统运行之前就确定了，如时间驱动的调度	具有确定性，但缺乏灵活性，适用于特征能够预先确定，且不容易发生变化的应用。
在线	调度信息则在系统运行过程中动态获得，如优先级驱动的调度（如EDF、 RMS 等）	灵活性高

嵌入式操作系统

抢占	正在运行的任务可能被其他任务打断。	实时内核大都采用了抢占式调度算法，较复杂
非抢占	一旦任务开始运行，该任务只有在运行完成而主动放弃CPU 资源，或是因为等待其他资源被阻 塞的情况下才会停止运行。	需要按照预先确定的顺序执行，且只有当任务主动放弃 CPU 资源后，其他任务才能得到执行的情况。



嵌入式操作系统

静态	所有任务的优先级在设计时已经确定下来，且在运行过程中不会发生变化（RMS）	适用于能够完全把握系统中所有任务及其时间约束（如截至时间、运行时间、优先顺序和 运行过程中的到达时间）特性的情况。静态调度比较简单，但缺乏灵活性，不利于系统扩展；
动态	任务的优先级则在运行过程中确定，并可能不断发生变化（如EDF)。	有足够的灵活性来处理变化的系统情况，但需要消耗更多的系统资源。

◆总结：
离线调度算法：系统运行前确定调度信息，如时间驱动，确定性，缺乏灵活性。
在线调度算法：系统运行中动态获得调度信息，如优先级驱动，灵活性较大。
抢占调度算法：运行任务可能被打断，更复杂，更耗资源。
非抢占调度算法：运行任务不被打断。
静态调度算法：任务优先级在设计时确定，不变化，简单，缺乏灵活性。
动态调度算法：任务优先级在运行中确定，不断变化，灵活，耗资源。

嵌入式操作系统

◆实时调度算法中还有强实时调度算法，具体可以分为：（★）

✓ 最早截止时间优先（Earliest Deadline First , EDF ）调度算法：根据任务截止时间确定优先级，截止时间越早，其优先级越高。

✓ 最低松弛度优先（Least Laxity First , LLF ）调度算法：根据任务紧急或松弛程度确定优先级，紧急程度越高，优先级越高。

✓ 单调速率（Rate Monotonic Scheduling, RMS)调度算法：根据任务周期确定有限期，周期越短，优先级越高。这种算法被认为是最优的。

最早截止时间优先算法既可用于剥夺式调度，也可用于非剥夺式调度方式中。

A和B都是周期性任务，A每隔20秒启动一次，每次执行10秒，B每隔50秒启动一次，每次执行25秒，试用固定优先级和抢占式EDF调度算法分析执行次序。

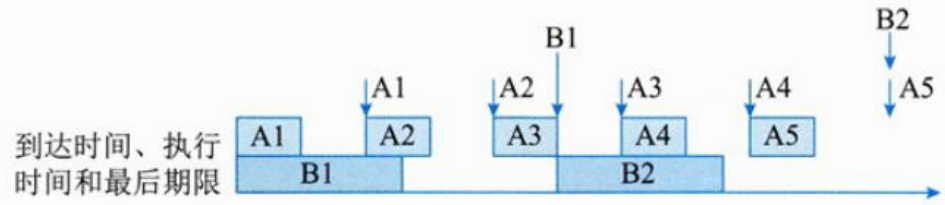
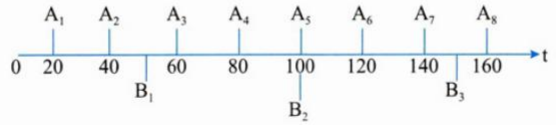


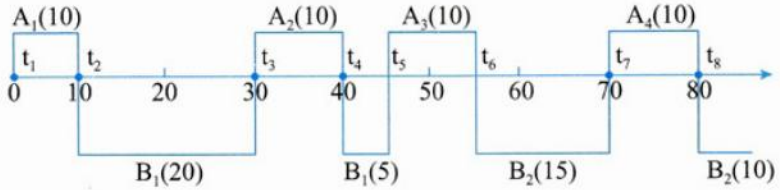
图 16-11　抢占的 EDF 调度算法示例

例如，在嵌入式实时系统中，有两个周期性实时任务 A 和 B，任务 A 要求每 20ms 执行一次，执行时间为 10ms；任务 B 则要求每 50ms 执行一次，执行时间为 25ms。任务 A 和 B 每次必须完成的时间 A1，A2，A3，…和 B1，B2，B3，…。如图 16-12（b）所示。



（a）

松弛度=必须完成时间-其本身的运行时间-当前时间



t=0ms时，A1的松弛度是10ms，B1的松弛度是25ms，调度A1执行；
t=10ms时，A2尚未到达，调度B1执行；
t=30ms时，A2的松弛度是0ms，B1的松弛度是15（50-5-30）ms，调度A2执行；
t=40ms时，A3的松弛度是10ms，B1的松弛度是5（50-5-40）ms，重新调度B1执行；
t=45ms时，B1执行完成，A3的松弛度是5（60-10-45）ms，调度A3执行；
t=55ms时，A尚未进入第4周期，B已进入第二周期，故再调度B2执行；
t=70ms时，A4松弛度已减至0ms，而B2的松弛度是20（100-70-10）ms，
故此时又应该抢占B2的处理机而调度A4执行。

（b）

图 16-12　LLF 调度原理示例

例如，我们有两个进程 P_1 和 P_2。P_1 和 P_2 的周期分别为 50ms 和 100ms，即 P_1=50ms 和 P_2=100ms。P_1 和 P_2 的处理时间分别为 t_1=20ms 和 t_2=35ms。每个进程的截止期限要求，它在下一个周期开始之前完成 CPU 执行。

使用单调速率调度，这里 P_1 分配的优先级要高于 P_2 的，因为 P_1 的周期比 P_2 的更短。在这种情况下，这些进程执行如图 16-13 所示。

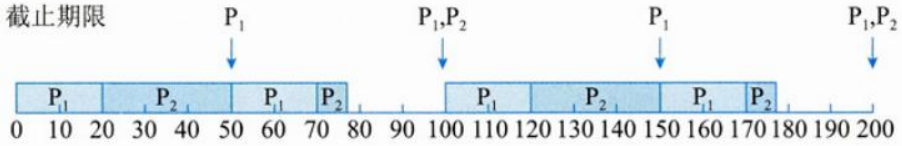


图 16-13　RMS 法原理示例

首先，P_1 开始，并在时间 20ms 完成 CPU 执行，从而满足第一个截止期限。P_2 在这点开始运行，并运行直到时间 50ms。此时，它被 P_1 抢占，尽管它的 CPU 执行仍有 5ms 的时间。P_1 在时间 70ms 完成 CPU 执行，在这点调度器恢复 P_2。P_1 在时间 75ms 完成 CPU 执行，也满足第一个截止期限。然后，系统一直空闲直到时间 100ms，这时，P_1 再次被调度。

单调速率调度可认为是最优的，因为如果一组进程不能由此算法调度，它不能由任何其他分配静态优先级的算法来调度。

嵌入式操作系统

◆存储管理

存储管理的主要目的是解决多个用户使用主存的问题，存储管理方法主要包括分区、分页、分段、段页式存储管理以及虚拟存储管理等。

◆任务间通信

任务间通信管理也是嵌入式操作系统的关键功能之一。它主要为操作系统的应用程序提供多种类型的数据传输、任务同步／异步操作等手段。

◆操作系统任务之间一般存在以下关系：相互独立、竞争、同步、通信。要实现多任务间的协同工作，操作系统必须提供任务间的通信手段。嵌入式操作系统常用的通信方式包括：（★）

- 共享内存：数据的简单共享。多任务访问同一地址空间。
- 信号量：基本的互斥和同步。最优，主要手段，任务间最快速通信。
- 消息队列：同一CPU内多任务间消息传递。类似于缓冲区的对象，像一个管道接收发送者消息等待接收者读取。
- Socket和远程调用：任务间透明的网络通信，不同计算机之间通信。
- Signals(信号):用于异常处理。通知进程发生了异步事件，是对中断机制的一种模拟。

嵌入式数据库

◆与传统数据库相比，嵌入式数据库系统有以下几个主要特点：嵌入式、实时性、移动性、伸缩性。
◆嵌入式数据库按存储位置的不同可分为三类：基于内存方式、基于文件方式、基于网络方式。（★★）

(1)基于内存的数据库系统(MMDB)是实时系统和数据库系统的有机结合。内存数据库是支持实时事务的最佳技术，其本质特征是以其“主拷贝”或“工作版本”常驻内存，即活动事务只与实时内存数据库的内存拷贝打交道。典型产品是eXtremeDB嵌入式数据库。eXtremeDB主要特点：
●最小化支持持久数据所必需的资源：实质上就是将内存资源减到最小。
●保持极小的必要堆空间：在某些配置上eXtremeDB只需要不到1KB的堆空间。
●维持极小的代码体积。
●通过紧密的集成持久存储和宿主应用程序语言消除额外的代码层。
●提供对动态数据结构的本地支持：例如变长字符串、链表和树。

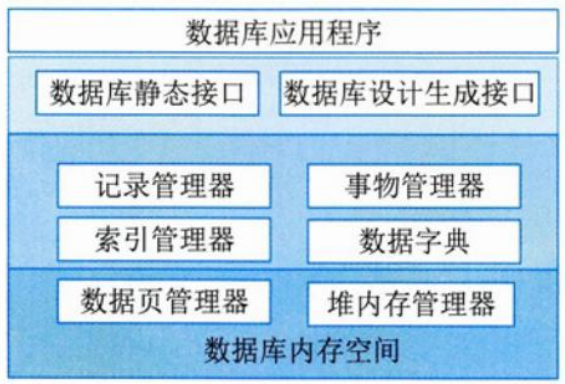


图 16-17　eXtremeDB 嵌入式数据库的架构

嵌入式数据库

(2)基于文件的数据库(FDB)系统就是以文件方式存储数据库数据，即数据按照一定格式储存在磁盘中。使用时由应用程序通过相应的驱动程序甚至直接对数据文件进行读写。典型产品是SQLite,它由公共接口、编译器系统、虚拟机和后端四个子系统组成。SQLite主要特点：（★）

● SQLite是一个开源的、内嵌式的关系型数据库。

● SQLite数据库服务器就在你的数据库应用程序中，其好处是不需要网络配置和管理，也不需要通过设置数据源访问数据库服务器。

● SQLite数据库的服务器和客户端运行在同一个进程中。这样可以减少网络访问的消耗，简化数据库管理，使你的程序部署起来更容易。

● SQLite在处理数据类型时与其他的数据库不同。区别在于它所支持的类型以及这些类型是如何存储、比较、强化(enforce)和指派(assign)。

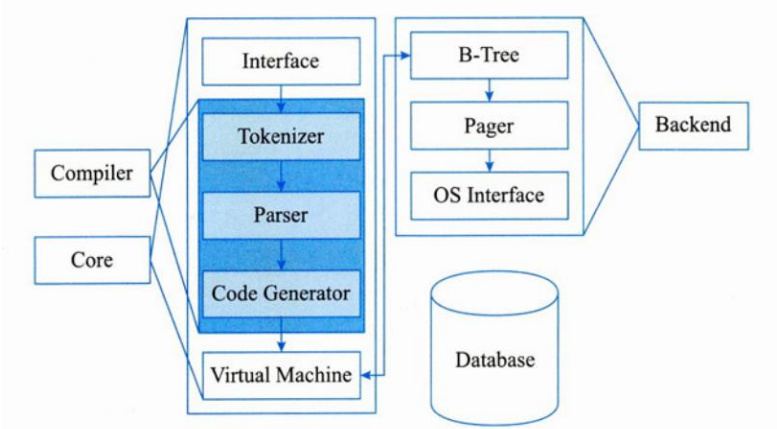


图 16-18　SQLite 嵌入式数据库系统架构

嵌入式数据库

(3)基于网络的数据库(NDB)系统是基于手机4G/5G的移动通信基础之上的数据库系统，在逻辑上可以把嵌入式设备看作远程服务器的一个客户端。实际上，嵌入式网络数据库是把功能强大的远程数据库映射到本地数据库，使嵌入式设备访问远程数据库就像访问本地数据库一样方便。
嵌入式网络数据库主要由三部分组成：客户端、通信协议和远程服务器。（★★）
其架构如下

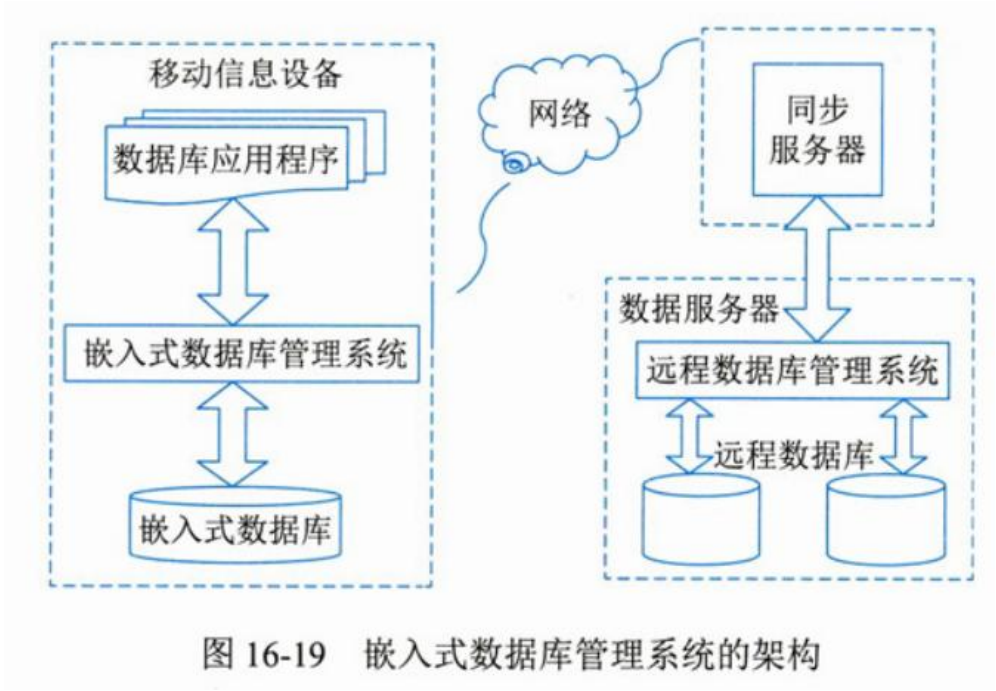


图 16-19　嵌入式数据库管理系统的架构

嵌入式数据库

◆数据库服务器架构：数据库客户端通常通过数据库驱动程序如JDBC、ODBC等访问数据库服务器，数据库服务器再操作数据库文件。数据库服务是一种客户端服务器模式，客户端和服务器是完全两个独立的进程。它们可以分别位于在不同的计算机甚至网络中。客户端和服务器通过TCP/IP进行通信。这种模式将数据与应用程序分离，便于对数据访问的控制和管理。
◆嵌入式数据库架构：嵌入式数据库不需要数据库驱动程序，直接将数据库的库文件链接到应用程序中。应用程序通过API访问数据库，而不是TCP/IP。因此，嵌入式数据库的部署是与应用程序在一起的。

◆数据库服务器和嵌入式数据库对比如下：（★）
(1)数据库服务器通常允许非开发人员对数据库进行操作，而在嵌入式数据中通常只允许应用程序对其进行访问和控制。
(2)数据库服务器将数据与程序分离，便于对数据库访问的控制。而嵌入式数据库则将数据的访问控制完全交给应用程序，由应用程序来进行控制。
(3)数据库服务器需要独立的安装、部署和管理，而嵌入式数据通常和应用程序一起发布，不需要单独地部署一个数据库服务器，具有程序携带性的特点。
◆嵌入式数据库有其自身的特殊需要，它应具备的功能包括以下4点：
●足够高效的数据存储机制；
●数据安全控制(锁机制)；
●实时事务管理机制；
●数据库恢复机制(历史数据存储)。

嵌入式数据库

◆一般嵌入式数据库可划分成数据库运行处理、数据库存取、数据管理、数据库维护和数据库定义等功能。（★）

◆在嵌入式环境下开发实时数据库系统需要特别解决以下几个设计问题：（★）
(1)存储空间管理模块。嵌入式实时数据库系统由于采用了内存缓冲的技术，必然要涉及嵌入式操作系统的内存管理。系统运行时，由该模块通过实时OS向系统申请内存缓冲区，作为共享的内存数据区使用。之后，将历史数据库中的初始化数据调入内存区对这些空白内存进行初始化。
(2)数据安全性、完整性控制模块。
(3)事务并发控制模块。在实时数据库中的封锁粒度通常选择一张关系表为一个单位。
(4)实时数据转储模块。该模块实现的功能是将实时数据存储为历史数据，通常由该模块先将历史数据保存在内存缓冲区中，缓冲区满时才一次性写入磁盘；读历史数据时，先从缓冲区内取数据，取不到数据时再进行文件的读写。
(5)运行日志管理模块。日志文件可以用来进行事务故障恢复和系统故障恢复。

嵌入式中间件

◆嵌入式中间件是在嵌入式系统中处于嵌入式应用和操作系统之间层次的中间软件，其主要作用是对嵌入式应用屏蔽底层操作系统的异构性，常见功能有网络通信、内存管理和数据处理等。（★）
◆中间件=平台+通信。中间件的共性特点：通用性、异构性、分布性、协议规范性、接口标准化。

◆具体到嵌入式中间件而言，它还应提供对下列环境的支持：
- 网络化：支持移动、无线环境下的分布应用，适应多种设备特性以及不断变化的网络环境；
- 支持流媒体应用，适应不断变化的访问流量和宽带约束；
- QoS质量品质：在分布式嵌入式实时环境下，适应强QoS的分布应用的软硬件约束；
- 适应性：能够适应未来确定的应用要求。

◆通用中间件大致存在以下几类：（★★）
- 企业服务总线中间件：ESB是一种开放的、基于标准的分布式同步/异步信息传递中间件。
- 事务处理监控器：为发生在对象间的事务处理提供监控功能，以保证操作成功。
- 分布式计算环境：指创建运行在不同平台上的分布式应用程序所需的一组技术服务。
- 远程过程调用：指客户机向服务器发送关于运行某程序的请求时所需的标准。
- 对象请求代理：为用户提供与其他分布式网络环境中对象通信的接口。
- 数据库访问中间件：支持用户访问各种操作系统或应用程序中的数据库。
- 消息传递：电子邮件系统是该类中间件的其中之一。
- 基于XML的中间件：XML允许开发人员为实现Internet中交换结构化信息而创建文档。

目录

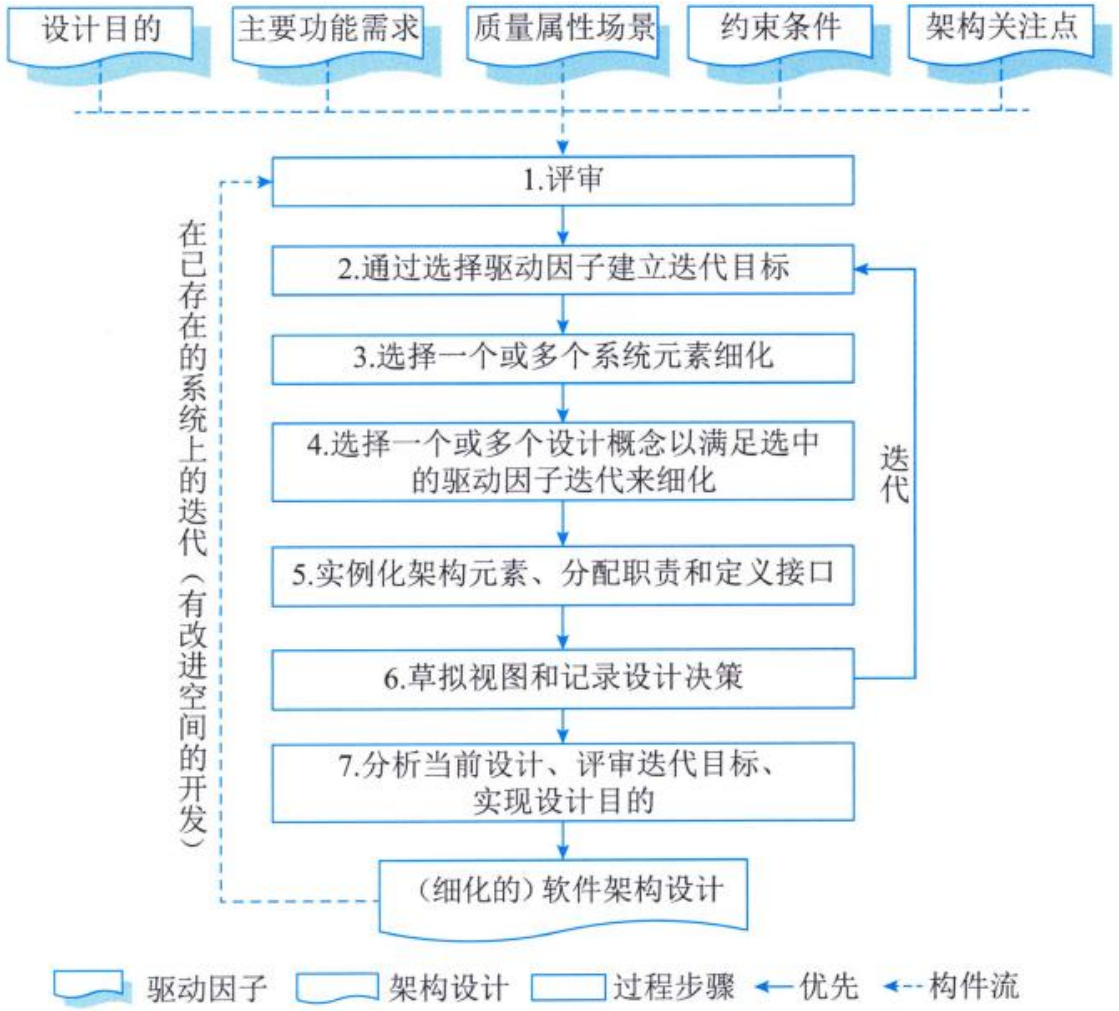
嵌入式系统软件架构设计方法

三种方法：ABSD、ADD和DARTS方法

一、在嵌入式系统中，其设计通常采用了自顶向下的设计方法，基于架构的软件设计(ABSD) 可适应于嵌入式系统的软件设计方法。

二、属性驱动的软件设计(ADD) 是把一组质量属性场景作为输入，利用对质量属性实现与架构设计之间的关系的了解(如体系结构风格、质量战术等)对软件架构进行设计的一种方法。

◆采用ADD方法进行软件开发时， 需要经历评审、选择驱动因子、选择系统元素、选择设计概念、实体化元素和定义接口、草拟视图和分析评价等七个阶段，如图所示：（★★）



嵌入式系统软件架构设计方法

步骤一：评审输入。确保设计流程的输入是可用且正确的。确认设计目的是否符合设计的类型。如果是设计一个已有的系统，还需要分析已经存在的架构设计的输入存在是否合理。

步骤二：通过选择驱动因子(架构)建立迭代目标。根据使用的开发模型去选择设计的回合。一个设计回合需要在一系列的设计迭代中进行，每一个迭代着重完成一个目标，特别是满足驱动因子的目标。

步骤三：选择一个或者多个系统元素来细化。选取可满足驱动因子需要的一个或者多个架构结构。

步骤四：选择一个或者多个设计概念来细化。从常用的架构设计模式中选取一种或多种设计概念，对选中的驱动因子进行细化。

步骤五：实例化架构元素、分配职责和定义接口。选择好了一个或者多个设计概念后，就要求做另一个设计决策了，包括所选择的实例化元素的设计概念。比如，如果选择分层，就需要决定分多少层。

步骤六：草拟视图和记录设计决策。将上述活动的结果用文字或图的方式记录或绘制出来。

步骤七：分析当前设计、评审迭代目标、实现设计目的。到本步骤，应该说已经创建好了部分设计，可以得到这个迭代设计建立的目标。在这个确定的目标前提下，可以得到项目利益相关者的认同，避免否定，导致返工。

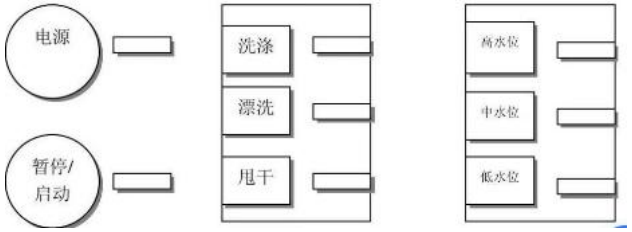
嵌入式系统软件架构设计方法-案例引入

功能需求定义和描述

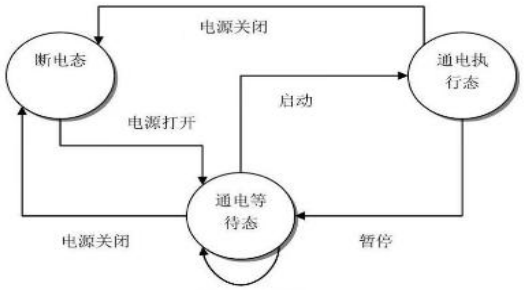
- 默认状态下洗衣机处于关闭状态。
- 电源开关开启后，洗衣机默认的程序功能是洗衣、漂洗、脱水，默认水位时中等水位，运行状态是等待状态；
- 洗衣机有运行状态和等待状态；
- 当洗衣机处于运行状态时，用户的程序控制，即洗衣、漂洗、脱水按键和水位按键的输入无效；只有当洗衣机处于等待状态时其输入才有效；
- 程序控制的洗衣、漂洗、脱水这三个键可以按下一个、两个或者三个，洗衣机总是按洗衣->漂洗->脱水的顺序执行程序并且只执行用户选择的程序；
- 水位按键每次只能选择高、中、低三种水位的一个值；
- 洗衣机显示面板显示洗衣机的电源状态、暂停\启动状态、当前执行的程序以及即将执行的程序和水位状态，并且能根据用户输入以后及时改变显示信息。

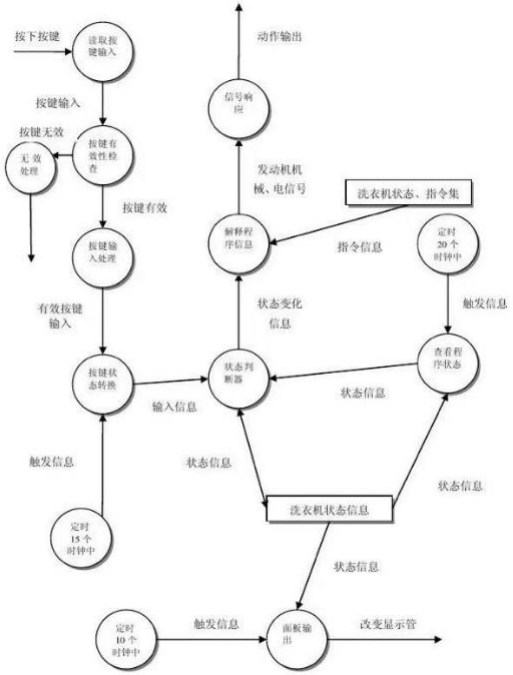
非功能需求定义和描述

- 洗衣机开关机的响应时间控制在500ms以内；
- 洗衣机在运行状态和等待状态之间的才换控制在800ms以内；
- 洗衣机功能和水位的输入控制在600ms以内；
- 洗衣机显示面板的响应控制在1200ms以内；
- 这里假设洗衣机洗衣用600s的时间，漂洗用300s的时间，脱水用100s的时间。



系统状态变迁图

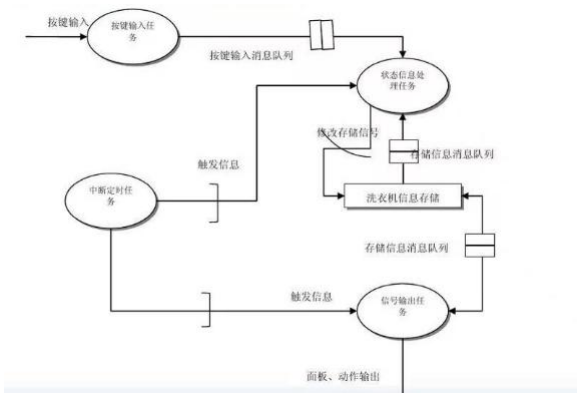


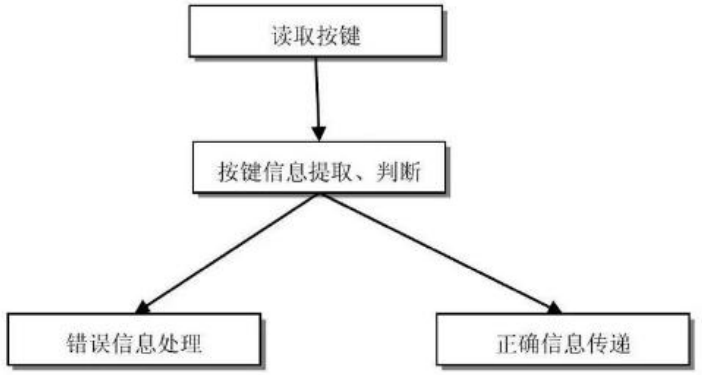


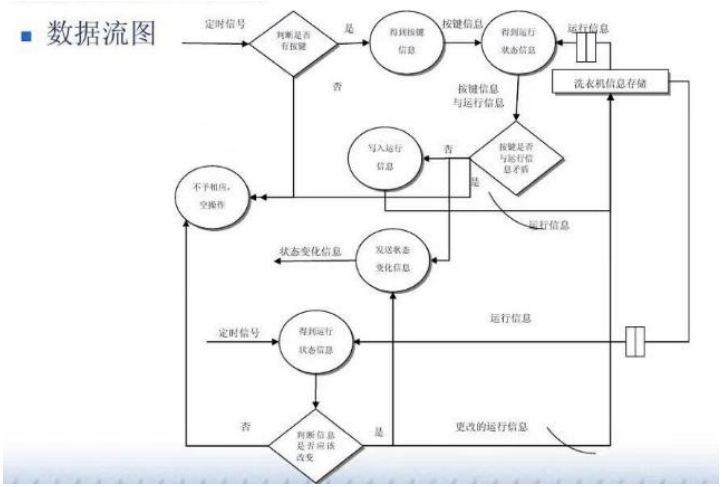
洗衣机控制软件任务划分(在数据流图中被划入相同的任务的数据处理过程用相同颜色的虚线方框来表示)

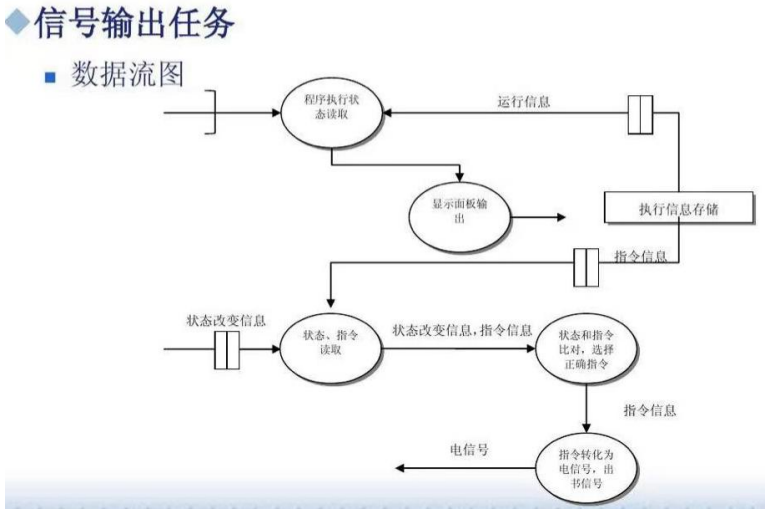
- 按键输入任务：功能内聚原则；
- 中断定时任务：周期执行原则；
- 洗衣机状态信息处理任务：计算需求和功能内聚原则；
- 信号输出任务：I/O依赖性原则。
- 为了提高效率，将数据流图中的洗衣机状态信息和洗衣机状态指令集整合在一起作为洗衣机程序的信息存储，以便各个任务的访问能够有一个统一的入口。

嵌入式系统软件架构设计方法-案例引入









嵌入式系统软件架构设计方法

三、实时系统设计方法（DARTS）主要是将实时系统分解为多个并发任务，并定义这些任务之间的接口。提供了一些分解规则和一套处理并发任务的设计步骤，还提供了一套把实时系统建造成并发任务的标准和定义并发任务间接口的指南。（★★）

◆ RTSAD（实时结构化分析和设计方法）是DARTS方法的起源，是对传统结构化分析和设计方法的补充扩展，专门用于开发实时系统。
◆实时结构化分析(RTSA) 主要对传统的结构化分析方法扩充了行为建模部分，它通过状态转换图(STD)刻画系统的行为特征，并利用控制转换与数据流图集成在一起。
◆实时结构化设计(RTSD) 是利用内聚和耦合原则进行程序设计的一个方法，它通过事务和变换两种策略将RTSA 的分析结构DFD/CFD 转换为程序结构图。（★★）

◆任务结构化标准可以为设计人员将实时系统分解为并发任务的时候提供帮助。这些标准是从设计并发系统所积累的经验中得到的启发。确定任务过程中主要考虑的问题是系统内部功能的并发特性。
信息隐藏作为封装数据存储的标准来使用。信息隐藏模块用于信息数据存储和状态转换表的内容和表示。
◆ RTSAD设计方法使用任务架构图来显示系统分解为并发任务的过程，以及采用消息、事件和信息隐藏模块形式的任务间接口。（★★）

嵌入式系统软件架构设计方法

◆ DARTS方法由以下5部分组成：（★★★）

1)用实时结构化分析方法(RTSA) 开发系统规范：本阶段要开发系统环境图(SCD) 和状态转换图(STD)。

2)将系统划分为多个并发任务：任务结构化标准应用于数据流/控制流图层次集事件合中的叶子节点上。初步任务架构图(TAD) 可以显示使用任务结构化标准确定的任务。

3)定义任务间接口：通过分析在上一阶段确认的任务间的数据/控制流接口可以定义任务间的接口。任务间的数据流被映射为松耦合的或紧密耦合的消息接口。事件流被映射为事件信号。数据存储被映射为信息隐藏模块。这个阶段应该完成时间约束分析。

4)设计每个任务：每个任务都代表了一个顺序程序的执行。在这个阶段要定义各个模块的功能以及与其他模块之间的接口。此外，还要设计各个模块的内部结构。

5)设计过程的成果：RTSA规范、任务结构规范（定义每个并发任务功能及接口）、任务分解（定义每个任务分解为模块的过程以及模块的功能接口详细设计）。

◆ DARTS开发方法的主要优势：

- 强调把系统分解成并发的任务，并提供了确认这些任务的标准；
- 提供了详细的定义任务间接口的指南；
- 强调了用任务架构图(STD) 的重要性，这在实时系统的设计中也非常重要；
- 提供了从RTSA规格到实时设计的转换。

◆ DARTS开发方法的不足之处：

- 用结构化的设计方法把任务创建成了程序模块，而并非完全用IHM来封装数据存储。
- 如果RTSA阶段的工作没有做好，创建任务就非常困难。

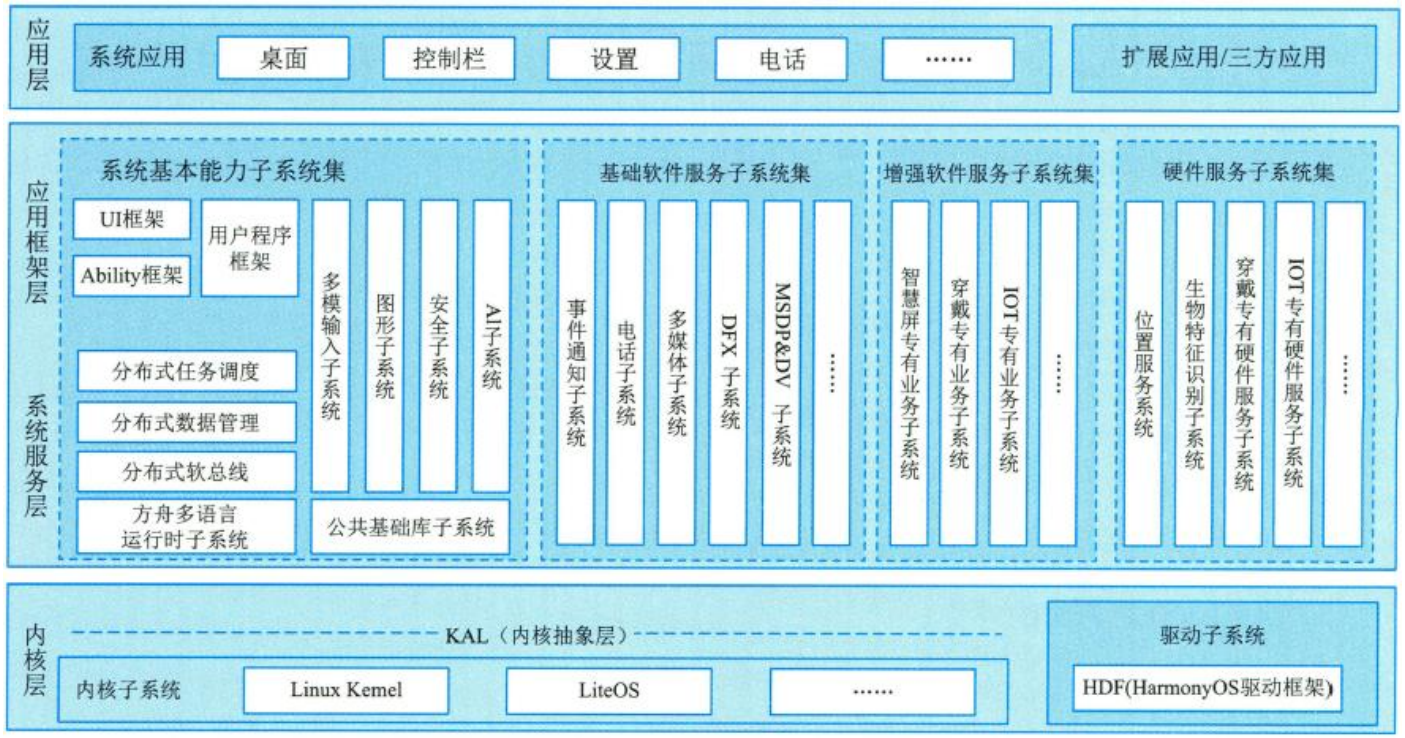
目录

嵌入式系统软件架构案例分析

◆鸿蒙操作系统架构案例分析

◆鸿蒙(HarmonyOS) 整体采用分层的层次化设计，从下向上依次为：内核层、系统服务层、框架层和应用层。（★★★）

系统功能按照“系统”→“子系统”→“功能/模块”逐级展开，在多设备部署场景下，支持根据实际需求裁剪某些非必要的子系统或功能/模块。



嵌入式系统软件架构案例分析

1)内核层：主要由内核子系统和驱动子系统组成。
◆内核子系统：HarmonyOS采用多内核设计，支持针对不同资源受限设备选用适合的OS内核。内核抽象层通过屏蔽多内核差异，对上层提供基础的内核能力。
◆驱动子系统：提供统一外设访问能力和驱动开发、管理框架。

2)系统服务层：是HarmonyOS 的核心能力集合，通过框架层对应用程序提供服务。
该层包含4个部分：
◆系统基本能力子系统集：为分布式应用在HarmonyOS多设备上的运行、调度、迁移等操作提供了基础能力。
◆基础软件服务子系统集：为HarmonyOS提供公共的、通用的软件服务。
◆增强软件服务子系统集：为HarmonyOS 提供针对不同设备的、差异化的能力增强型软件服务。
◆硬件服务子系统集：为HarmonyOS 提供硬件服务。

3)框架层：为HarmonyOS 的应用程序提供了Java/C/C++/JS 等多语言的用户程序框架和Ability框架，以及各种软硬件服务对外开放的多语言框架API; 同时为采用HarmonyOS 的设备提供了C/C++/JS 等多语言的框架API, 不同设备支持的API 与系统的组件化裁剪程度相关。

4)应用层：包括系统应用和第三方非系统应用。HarmonyOS 的应用由一个或多个FA(Feature Ability) 或PA(Particle Ability) 组成。其中， F A有U I 界面，提供与用户交互的能力；而PA无U I 界面，提供后台运行任务的能力以及统一的数据访问抽象。

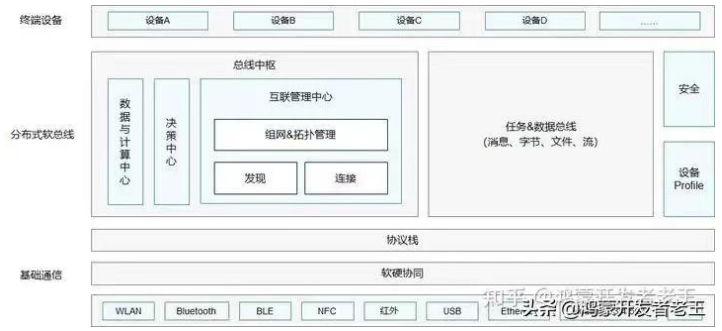
嵌入式系统软件架构案例分析

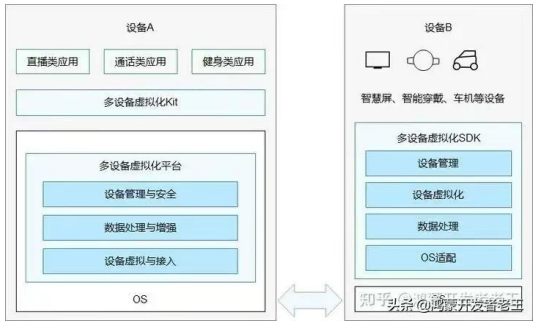
◆鸿蒙操作系统架构具有4个技术特性：（★★★）
1)分布式架构首次用于终端OS, 实现跨终端无缝协同体验。
2)确定时延引擎和高性能IPC技术实现系统天生流畅。
3)基于微内核架构重塑终端设备可信安全。
4)通过统一IDE支撑一次开发，多端部署，实现跨终端生态共享。

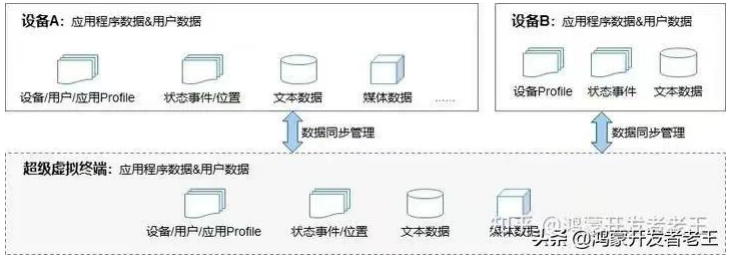
◆ HarmonyOS 架构的系统安全性主要体现在搭载HarmonyOS 的分布式终端上，可以保证“正确的人，通过正确的设备，正确地使用数据”。通过“分布式多端协同身份认证”来保证“正确的人”,通过“在分布式终端上构筑可信运行环境”来保证“正确的设备”,通过“分布式数据在跨终端流动的过程中，对数据进行分类分级管理”来保证“正确地使用数据”。（★★）

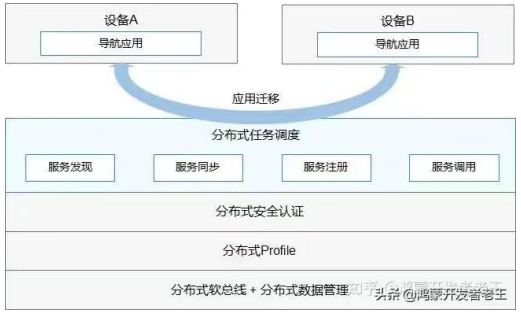
嵌入式系统软件架构案例分析

在HarmonyOS架构中，重点关注于分布式架构所带来的优势，主要体现在下面四个方面：（★★★）
◆分布式软总线是多种终端设备的统一基座，为设备之间的互联互通提供了统一的分布式通信能力；
◆分布式设备虚拟化平台可以实现不同设备的资源融合、设备管理、数据处理，多种设备共同形成一个超级虚拟终端。针对不同类型的任务，为用户匹配并选择能力合适的执行硬件；
◆分布式数据管理基于分布式软总线的能力，实现应用程序数据和用户数据的分布式管理。用户数据不再与单一物理设备绑定，业务逻辑与数据存储分离，应用跨设备运行时数据无缝衔接；
◆分布式任务调度构建统一的分布式服务管理(发现、同步、注册、调用)机制，支持对跨设备的应用进行远程启动、远程调用、远程连接以及迁移等操作，选择合适的设备运行分布式任务。









嵌入式系统软件架构案例分析

面向安全攸关系统的跨领域GENESYS系统架构案例分析

◆GENESYS是一种跨领域的通用嵌入式架构平台，主要解决了嵌入式系统面临的三方面挑战：复杂性管理的挑战(采用消息交换方式提高软硬件抽象级别)、系统健壮性的挑战(设计故障的隔离框架)、能量有效使用的挑战(采用综合化资源管理方法)。

◆GENESYS整个架构包括两类构成系统：即构件和基础平台。基础平台提供了一种“腰”型核心服务和大量用于实现系统构件的可选择服务的最小集合。

◆GENESYS架构主要提供了三组服务，即领域无关服务、领域专用服务和应用专用服务(包括中间件)。如图所示：（★★）

(1)核心服务。是强制性的，是GENESYS架构实例的一部分。核心服务应包含那些可构造较高级服务或者为了维持该结构性质而不可缺少的服务，它是系统服务中的最小集。

(2)选择服务。是在核心服务之上构造的。它是一种需要时可以扩展的开放式的集合。

(3)领域专用服务。是由领域特有的选择服务子集加上待开发的领域特征的特定服务组合。

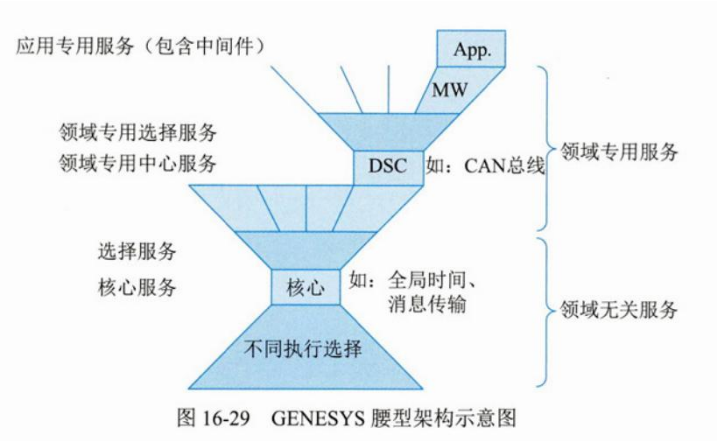


图 16-29　GENESYS 腰型架构示意图

嵌入式系统软件架构案例分析

◆ GENESYS架构的重要思想是分离计算与通信，将计算构件和通信设施作为独立构件进行设计。
◆ GENESYS的通信设施构件是基于消息传输的风格。构件中的基本交往机制是多播单向消息的交换。消息在发送时刻发出，在某个稍后的时刻达到在接收者那里。每一个消息有专门标识的发送者和若干个接受者。
◆ GENESYS架构将构件归为四类：硬件构件和软件构件、系统构件和应用构件。硬件构件的功能使用硬件被预先确定，因此不能修改。在软件构件中，将加载在软件构件上的软件称为作业。将作业分配给适当的可以执行该作业的硬件单元就创建了新的构件。系统构件是提供某些架构服务的构件。系统构件可以广泛重用在许多不同的应用场景中，应用设计者只考虑应用构件的开发。

◆基于GENESYS架构的四类基于消息的构件接口：
◆链接接口(LIF):LIF提供了构件与构件之间基于消息的操作服务，它是构件的综合接口。
◆局部接口(LI):LI是构件连接到外部环境的接口，它建立了构件和局部环境之间的连接关系。
◆技术无关接口(TII):TII是指用于系统运行需要的配置或管理资源的接口，它属于非功能属性范畴。
◆技术相关的接口(TDI):TDI是指用于查看构件内部、观察构件的内部变量的接口，如构件诊断。

◆ GENESYS定义了三级的集成，即芯片级(Chip Level)、设备级(Device Level)和系统级(SystemLevel)。芯片级的构件是IP核，IP核间可通过NoC(Network of Chip)相互连接；设备级的构件是芯片，芯片间可以由内部通信芯片互相连接。一个设备可以是在互联网上的一个可寻址实体，也可以是一个IP地址；系统级的构件是设备，它们可以由有线或无线通信服务互相连接。

嵌入式系统软件架构案例分析

◆物联网操作系统软件架构

◆物联网操作系统通常包括了芯片层、终端层、边缘层、云端层等多个层面内容。（★）

◆物联网操作系统使用的软件以及技术主要有：
① 开源物联网操作系统 FreeRTOS
② 公共服务组件（网络协议、外设支持、可移植操作系统接口 POSIX 等）
③ 定制性服务组件：（消息队列遥测传输协议 MQTT，安全超文本传输协议 HTTPS，加密消息标准 PKCS #11 支持，安全套件等）

◆物联网操作系统主要特征有：内核实时性、内核尺寸伸缩性、架构可扩展性、高可靠性、低功耗。

典型真题

以下关于鸿蒙操作系统的叙述中，不正确的是（ ）。
A.鸿蒙操作系统整体架构采用分层的层次化设计，从下向上依次为：内核层、系统服务层、框架层和应用层
B.鸿蒙操作系统内核层采用宏内核设计，拥有更强的安全特性和低时延特点
C.鸿蒙操作系统架构采用了分布式设计理念，实现了分布式软总线、分布式设备系统的虚拟化、分布式数据管理和分布式任务调度等四种分布式能力
D.架构的系统安全性主要体现在搭载 HarmonyOS 的分布式终端上，可以保证“正确的人，通过正确的设备，正确地使用数据”
解析：鸿蒙操作系统采用微内核架构，整体采用层次式架构，采用分布式理念且实现了分布式安全框架。
答案：B

本章重点回顾

1、嵌入式系统特点
2、嵌入式软件架构方法
3、鸿蒙操作系统

THANKS