**总结**

一、考试形式：闭卷，笔试

基本概念、基本原理、设计应用技术

范围：以课件涵盖内容为主。

二、题型：

1. 简答题

2. 问答题

3. 设计题

三、复习

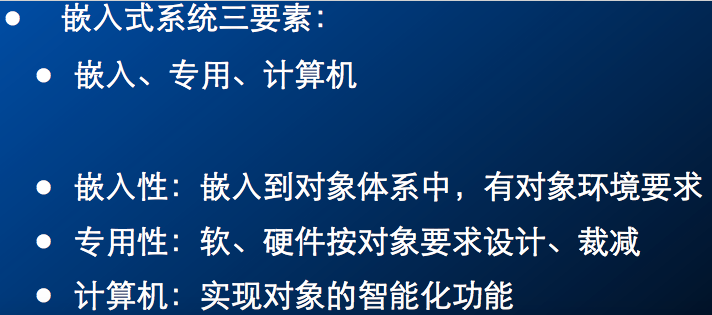
1. 嵌入式系统的定义，特点，分类，典型应用。\*\*

**定义：**嵌入式系统是“用于控制、监视或者辅助操作机器和设备的装置”

国内普遍被认同的定义:

嵌入式系统是“以应用为中心,以计算机技术为基础,软硬件可裁减,适用于应用系统对功能、可靠性、成本、体积、功耗有严格要求的专用计算机系统”。

嵌入式系统就是一个具有特定功能或用途的隐藏在某种设备中的计算机软硬件 集合体,没有固定的特征形状。



**特点：**嵌入式系统通常是形式多样、面向特定应用的

嵌入式系统得到多种类型的处理器和处理器体系结构的支持   
 嵌入式系统通常极其关注成本   
 嵌入式系统有实时性和可靠性的要求   
 嵌入式系统使用的操作系统一般是适应多种处理器、可剪裁、轻 量型、实时可靠、 可固化的嵌入式操作系统   
 嵌入式系统开发需要专门工具和特殊方法   
**分类：**

按嵌入式处理器的位数来分类

▪ 4位嵌入式系统▪ 8位嵌入式系统 ▪ 16位嵌入式系统▪ 32位嵌入式系统 ▪ 64位嵌入式系统

按应用来分类  
 ▪ 信息家电 ▪ 移动终端 ▪ 工业控制 ▪ 汽车电子 ▪ 通信类

按速度分类  
 ▪ 强实时系统, 其系统响应时间在毫秒或微秒级。

▪  一般实时系统, 其系统响应时间在几秒的数量级上,其实时性的要求比强实时系统要差一些。

▪  弱实时系统, 其系统响应时间约为数十秒或更长。这种系统的响应时间可能 随系统负载的轻重而变化。

按确定性来分类

根据确定性的强弱,可将嵌入式系统分为硬实时、软实时系统:

▪  硬实时:系统对系统响应时间有严格的要求,如果系统响应时间不能满足, 就要引起系统崩溃或致命的错误。

▪  软实时:系统对系统响应时间有要求,但是如果系统响应时间不能满足,不 会导致系统出现致命的错误或崩溃。

按嵌入式系统软件复杂程度来分类

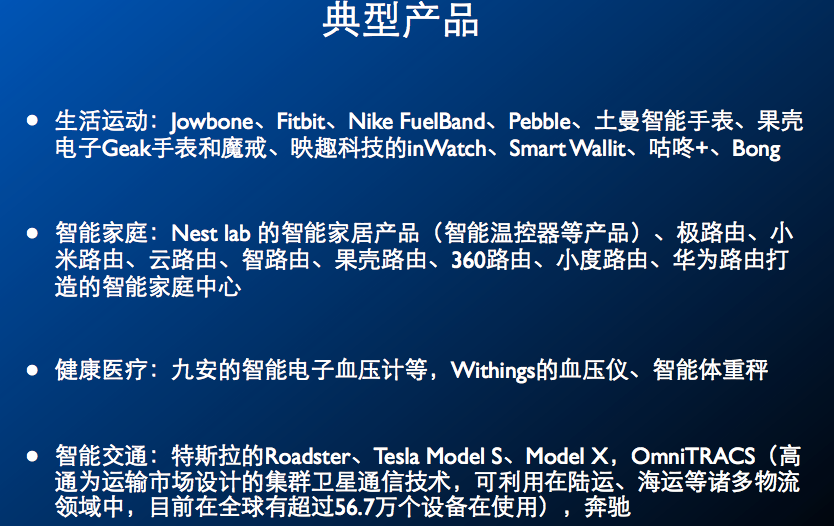
▪ 循环轮询系统

▪ 有限状态机系统

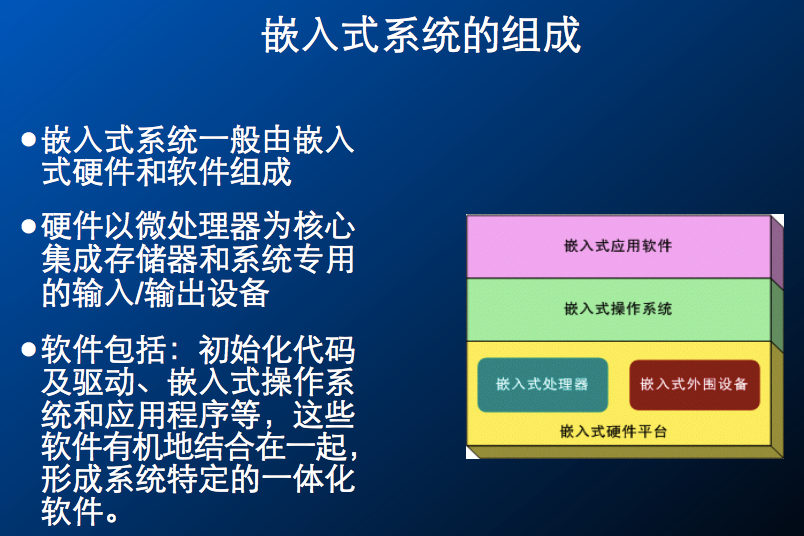
▪ 前后台系统

▪ 单处理器多任务系统

▪ 多处理器多任务系统

**典型应用：**

1. 嵌入式系统、CPS的基本组成\*。



1. 嵌入式系统设计\*

嵌入式系统面临挑战

1)需要多少硬件?

2)如何满足时限要求,如何处理多项功能在时间上的协调一致关系?

3)如何降低系统的功耗?

4)如何设计以保证系统可升级?

5)如何保证系统可靠地工作?

传统开发过程与软硬件协同设计

嵌入式系统的设计过程的基本流程：

需求分析

规格说明

体系结构

构件设计

系统集成

软硬件划分

嵌入式系统中软件和硬件协同完成系统的功能

软件硬件划分通常由速度、灵活性以及开销来决策

嵌⼊式系统的设计涉及硬件与软件部件,设计中必须决定什么功能由硬件实现,什么功能由软件实现。

硬件和软件具有双重性

软硬件变动对系统的决策造成影响

划分和选择需要考虑多种因素

硬件和软件的双重性是划分决策的前提

1. 嵌入式硬件系统基础。

嵌入式系统的硬件是以嵌入式微处理器为核心,主要由嵌入式微处理器、总线、存储器、 输入/输出接口和设备组成。

• 嵌入式微处理器  
• 总线  
• 存储器  
• 输入/输出接口和设备

4.1 嵌入式微处理器基础

4.1.1嵌入式微处理器体系结构

冯诺伊曼结构与哈佛结构

嵌入式微处理器体系结构： 诺伊曼体系结构和哈佛体系结构。

* + 1. 冯诺依曼体系结构数据和程序放在同一个存储单元，统一编址，指令和数据通过同一个总线访问。
    2. 哈佛体系结构的程序和数据不是放在同一个存储空间中，因此有两条总线，也就是说数据吞吐率是冯诺依曼结构的两倍。

CISC与RISC

|  |  |  |
| --- | --- | --- |
|  | CISC | RISC |
| 价格 | 由硬件完成部分软件功能，硬件复杂性增加，芯片成本高 | 有软件完成部分硬件功能，软件复杂性增加，芯片成本低 |
| 性能 | 减少代码尺寸，增加指令的执行周期数 | 使用流水线降低指令的执行周期数，增加代码尺寸 |
| 指令集 | 大量的混杂指令集，有简单快速的指令，也有复杂多周期指令，符合HLL | 简单的但周期指令，在汇编指令方面有相应的CISC微代码指令 |
| 高级语言支持 | 硬件完成 | 软件完成 |
| 寻址模式 | 复杂的寻址模式，支持内存到内存寻址 | 简单的寻址模式，仅允许LOAD和STORE指令存取内存，其它所有的操作都基于寄存器到寄存器 |
| 寄存器数目 | 寄存器较少 | 寄存器较多 |

复杂指令集系统CISC（Complex Instruction Set Computer）和精简指令集系统RISC（Reduced Instruction Set Computer）

流水线技术

RISC机器用来减少指令周期的一种技术，可提高处理器和总线的使用率；

信息存储的字节顺序（大端和小端）

字节排序按分为大端和小端，概念如下

大端(big endian):低地址存放高有效字节

小端(little endian):低字节存放低有效字节

0x12345678   16进制，两个数就是一字节

高有效字节——>低有效字节： 12 34 56 78

          低地址位     高低址位

**大端**：  12  34        56   78

**小端**： 78  56        34   12

4.1.2 ARM

工作状态

ARM微处理器的两种工作状态

ARM状态：

A、处理器执行32位的ARM指令

B、ARM指令要求字对齐

Thumb状态：

A、处理器执行16位的Thumb指令

B、Thumb指令要求半字对齐

运行模式

（1）用户模式(usr)： ARM处理器正常的程序执行状态

（2）系统模式(sys) ：运行具有特权的操作系统任务

（3）快中断模式(fiq) ：支持高速数据传输或通道处理

（4）管理模式(svc) ：操作系统保护模式

（5）数据访问终止模式(abt)：用于虚拟存储器及存储器保护

（6）中断模式(irq) ：用于通用的中断处理

（7）未定义指令终止模式(und)： 支持硬件协处理器的软件仿真

除用户模式外，其余6种模式称为非用户模式或特权模式；用户模式和系统模式之外的5种模式称为异常模式。ARM处理器的运行模式可以通过软件改变，也可以通过外部中断或异常处理改变。

寄存器结构

ARM处理器共有37个寄存器，均为32位，被分为若干个组（BANK），这些寄存器包括：

（1）31个通用寄存器，包括程序计数器（PC指针）。

R0-R15可见，其他用于加快异常处理过程；

R13：Stack Pointer（SP）

R14：Link Register（LR）

R15：Program Counter（PC）

（2）6个状态寄存器，用以标识CPU的工作状态及程序的运行状态，只使用了其中的一部分。

程序状态寄存器（CPSR）current program status register

程序状态保护寄存器（SPSR）saved program status register

中断和异常\*

中断的进入：

        保存CPSR位

        设置CPSR位（异常则设置异常状态）

        保存PC到LR中

        PC根据向量表设置对应的地址

中断的退出：

        CPSR的恢复

        PC的恢复

WatchDog\*

（1）看门狗的功能及工作原理

嵌入式控制系统运行时受到外部干扰或者系统错误，程序有时会出现“跑飞”，导致整个系统瘫痪。为了防止这一现象的发生，在对系统稳定性要求较高的场合往往要加入看门狗(Watchdog)电路。看门狗电路的作用就是当系统“跑飞”而进入死循环时，恢复系统的运行。

看门狗电路的基本原理为：设本系统程序完整运行一周期的时间是tp，看门狗的定时周期为ti，且ti>tp，在程序运行一周期后就修改（再重新设定看门狗的定时周期）定时器的计数值（俗称“喂狗”)，只要程序正常运行，定时器就不会溢出。若由于干扰等原因使系统不能在tp时刻修改定时器的计数值，定时器将在ti时刻溢出，引发系统复位，使系统得以重新运行，从而起到监控作用。

（2）看门狗定时器寄存器

* 看门狗定时器控制寄存器WTCON

通过该寄存器，可以使能/禁止看门狗、选择输入时钟源、使能/关闭中断、使能/关闭输出。该寄存器及其控制位的定义如表12-1和12-2所列。

* 看门狗定时器数据寄存器WTDAT

该数据寄存器用于设置看门狗定时器的初值。在初始的操作中，该值不会自动加载到定时器中，首次定时器初始值是0x8000，以后该寄存器的值会被自动加载到WTCNT寄存器中。一般是该定时器工作在通用定时器模式下时使用。

* 看门狗定时器/计数器寄存器WTCNT

该寄存器为看门狗定时器的计数器，它的值表示该定时器的当前计数值，即到下一次溢出还需要经历的时钟数。当该定时器工作在看门狗模式时使用该寄存器，每次溢出前需要重新设置其值，以防止发生复位。

4.2 嵌入式系统的存储体系

4.2.1存储器系统：存储器系统的层次结构

1.（CPU寄存器）寄存器组

2.（芯片内高速缓存）片内cache、写缓存、TCM、片内SRAM

3.（芯片外高速缓存）板卡级SRAM、DRAM和SRAM

4.（主存储器）FLASH PROM EPROM

5.（外部存储器）硬盘驱动器和光盘驱动器

4.2.2 ROM的种类与选型

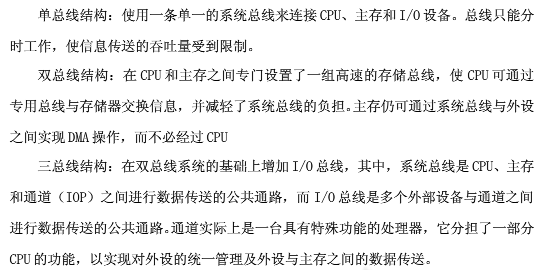
4.2.3 Flash的种类与选型

4.2.4 RAM的种类与选型

4.2.5 Cache-

4.3 嵌入式系统总线

4.3.1 总线结构：单、双、多



4.3.2 输入输出编程：忙等IO和中断IO

忙等效率比较低，CPU在等待设备的时候不能做其他工作，无法同步进行输入输出。

中断IO允许设备更改CPU控制流。

4.3.3 GPIO接口基本原理与结构（不考）

4.4嵌入系统通信及网络接口（不考）

Ref: 嵌入式计算系统设计原理。

1. 嵌入式系统软件知识

5.1嵌入式软件基础知识

5.1.1嵌入式软件的分类（系统软件、支撑软件、应用软件）

5.1.2嵌入式软件体系结构\*

（1）驱动层

– 板级初始化程序

– 与系统软件相关的驱动

– 与应用软件相关的驱动

– 与应用软件相关的驱动不一定需要与操作系统连接，这些驱动的设计和开发由应用决定。

（2）操作系统层

– 操作系统层包括嵌入式内核、嵌入式TCP/IP网络系统、嵌入式文件系统、嵌入式GUI系统和电源管理等部分。

– 其中嵌入式内核是基础和必备的部分，其他部分要根据嵌入式系统的需要来确定。

（3）中间件层

– 目前在一些复杂的嵌入式系统中也开始采用中间件技术，主要包括嵌入式CORBA、嵌入式Java、嵌入式DCOM和面向应用领域的中间件软件。

– 如基于嵌入式CORBA的应用于软件无线电台的应用中间件SCA（Software Core Architecture）等。

（4）应用层

– 应用层软件主要由多个相对独立的应用任务组成

– 每个应用任务完成特定的工作，如I/O任务、计算的任务、通信任务等，由操作系统调度各个任务的运行。

5.2 嵌入式操作系统基础知识

5.2.1 RTOS概念、特点、选型原则（商业化RTOS）\*

RTOS定义：对外来事件能在限定的响应时间内做出预定质量处理的计算机系统

特点：可移植性、强调实时性能、内核精简、抢占式内核、使用可重入函数、可配置、可裁剪、高可靠性。

举例：μC/OS-II，ThreadX，VxWorks，Linux/RT，FreeRTOS

选型原则：综合权衡

1.成本

2.可靠性

3.实时性

4.工具链

5.模块丰富

6.RTOS 内核 RAM、ROM 占用量

7.支持

5.2.2 任务管理

进程、线程、任务的概念

任务的实现（任务的层次结构、任务控制块、任务的状态及状态转换、任务队列）

任务调度（可抢占调度、不可抢占调度、先来先服务、时间片轮转算法、优先级算法）

实时系统及任务调度（RMS、EDF算法）\*\*

**对于下面的进程**

**进程 执行时间 期限**

**P1     1             3**

**P2     1             4**

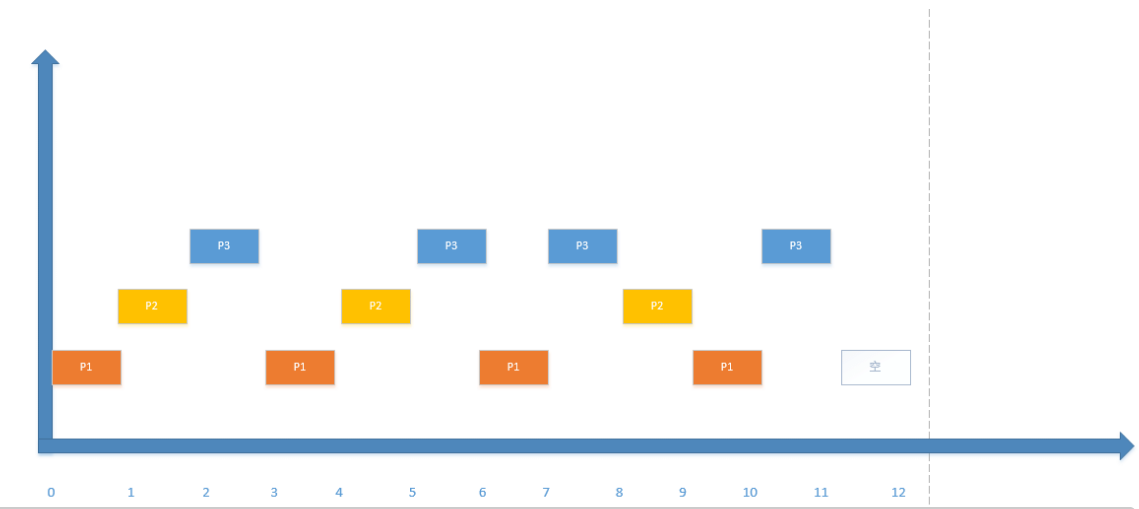
**P3     2             6**

**A、 用RMS策略调度这些进程**

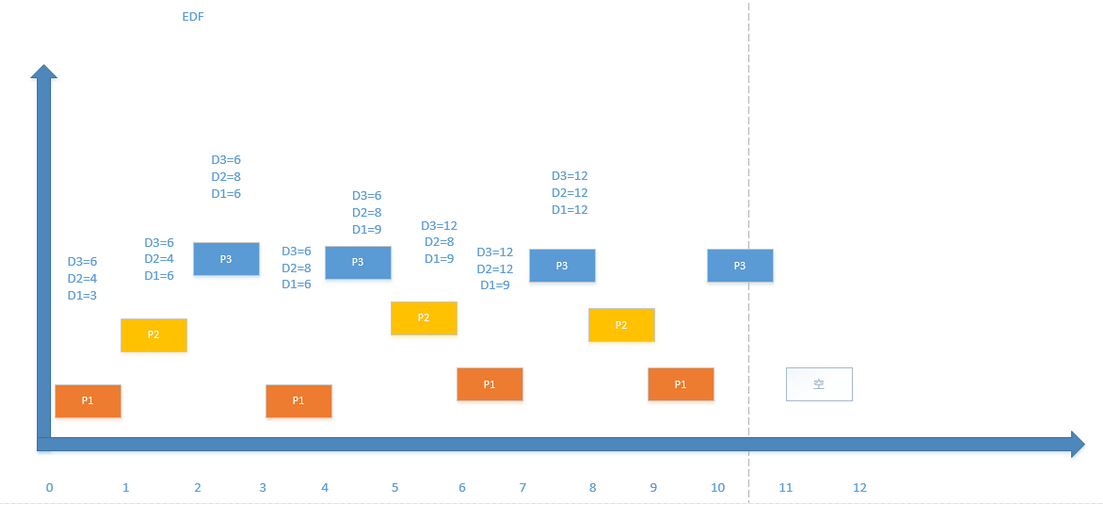
**B、 用EDF策略调度这些进程**

**在每种情况下 计算时间间隔等于进程周期最小公倍数的调度，时间从t=0开始**

A.



B、



任务间通信（共享内存、消息、管道、信号）

同步与互斥（竞争条件、临界区、互斥、信号量、死锁）

5.2.3存储管理（结合ucOSII）

5.2.4设备管理（不考）

设备无关性、I/O地址、I/O控制、中断处理

5.2.5文件系统基础知识（不考）

5.2.6操作系统移植基础知识（不考）

5.3 嵌入式系统程序设计（不考）

5.3.1嵌入式软件开发基础知识

5.3.2嵌入式程序设计语言

汇编、编译、解释系统的基础知识和基本工作原理

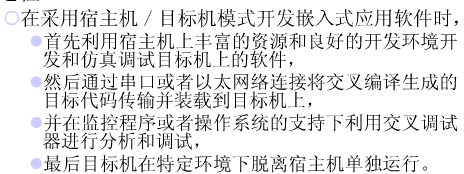
汇编语言

各类程序设计语言的主要特点和适用情况

5.3.3嵌入式软件开发环境

交叉开发（宿主机、目标机）\*

采取宿主机和目标机的模式，软件开发在宿主机（通常是PC）上，目标机用于运行软件。



编辑器、编译器、链接器、调试器、模拟器

常用嵌入式开发工具

集成开发环境

开发辅助工具

5.3.4嵌入式软件开发

Ref:

嵌入式计算系统设计原理。

嵌入式系统导论-CPS方法。

1. ucOS-II\*\*

任务调度\*\*、中断与时钟、同步与通信、存储管理（静态、动态）

* 1. 任务调度：

μC/OS\_II进行任务调度的思想是 “近似地每时每刻总是让优先级最高的就绪任务处于运行状态” 。为了保证这一点，它在系统或用户任务调用系统函数及执行中断服务程序结束时总是调用调度器，来确定应该运行的任务并运行它 。

μC/OS\_II进行任务调度的依据就是任务就绪表

* 1. 中断与时钟：
     1. μC/OS-II系统响应中断的过程为：
     2. 系统接收到中断请求后，这时如果CPU处于中断允许状态（即中断是开放的），系统就会中止正在运行的当前任务，而按照中断向量的指向转而去运行中断服务子程序；当中断服务子程序的运行结束后，系统将会根据情况返回到被中止的任务继续运行或者转向运行另一个具有更高优先级别的就绪任务。
  2. 同步与通信：

管理的核心是事件控制块ECB，同步可以通过开关中断实现。互斥使用信号量。

通信可以使用邮箱和消息队列实现。

* 1. 存储管理

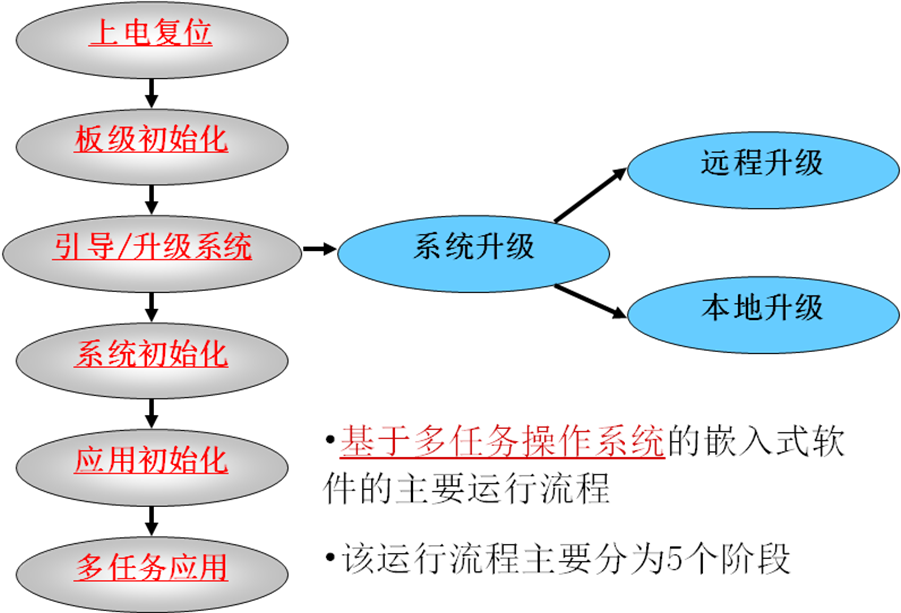
ucos-II不划分内核和用户空间，系统只有一个地址空间。

管理：多个内存分区，每个分区的块大小相同，不同分区的块大小不同，为了解决碎片问题。

1. bsp, bootloader\*

嵌入式软件运行流程

* + 1. 上电复位
    2. 板级初始化
    3. 引导操作系统
    4. 操作系统初始化
    5. 应用程序初始化
    6. 多任务应用



bsp，特点，与bios，EFI区别

* 1. BSP的概念：BSP 全称“板级支持包”(Board Support Packages),说的简单一点,就是一段启动代码, 和计算机主板的 BIOS 差不多,但提供的功能区别就相差很大。
  2. 特点与功能：
     1. 硬件相连性
        1. BSP必须为操作系统提供操作和控制具体硬件的方法；
     2. 操作系统相连性
        1. BSP必须为不同的操作系统提供接口；

区别：

BIOS主要是负责在电脑开启时检测、初始化系统设备（设置栈指针，中断分配，内存初始化..）、装入操作系统并调度操作系统向硬件发出的指令。

BSP是和操作系统绑在一起运行在主板上的，尽管BSP的开始部分和BIOS所做的工作类似，可是大部分和BIOS不同，作用也完全不同。

程序员还可以编程修改BSP，在BSP中任意添加一些和系统无关的驱动或程序，甚至可以把上层开发的统统放到BSP中。 而BIOS程序是用户不能更改，编译编程的，只能对参数进行修改设置。更不会包含一些基本的硬件驱动。

由于EFI框架比BIOS要大得多，其启动过程也比BIOS要复杂。于BIOS最大的区别就是EFI首先需要EBC虚拟机，然后再启动设备驱动和EFI应用程序，最后通过EFI boot manager加载操作系统引导程序。

rtos的引导模式

操作系统引导概念：将操作系统装入内存并开始执行的过程。

按时间效率和空间效率不同的要求，分为两种模式：

1.需要BootLoader的引导模式：节省空间，牺牲时间，适用于硬件成本低，运行速度快，但启动速度相对慢

2.不需要BootLoader的引导模式：时间效率高，系统快速启动，直接在NOR flash或ROM系列非易失性存储介质中运行，但不满足运行速度的要求

bootloader及其启动过程

bootloader的概念：bootloader是嵌入式系统中的OS启动加载程序。

启动过程：

* 初始化硬件，如设置UART（至少设置一个），检测存储器等
* 设置启动参数，告诉内核硬件的信息，如用哪个启动界面，波特率
* 跳转到操作系统的首地址
* 消亡

Ref：嵌入式实时操作系统的设计与开发。廖勇，电子工业出版社，2015.

1. 建模\*\*

有限状态机及其应用。

有限状态机的实现。

Ref：

嵌入式系统导论-CPS方法。

Chapter3, Practical UML Statecharts in C/C++,Second Edition.

Embedded System Design：Embedded System Foundations of Cyber-Physical Systems。

Further Readings：

Embedded System Design：Embedded System Foundations of Cyber-Physical Systems。

高性能嵌入式计算。Wayne Wolf著，机械工业出版社。

嵌入式系统设计与实践。Elecia White，机械工业出版社。

嵌入式软件开发精解。Colin Walls，机械工业出版社。

嵌入式实时操作系统的设计与开发。廖勇，电子工业出版社，2015.

ARM快速嵌入式系统原型设计 – 基于开源硬件mbed。

Arduino权威指南。Michael Margolis著，人民邮电出版社。

ARM嵌入式系统开发 – 软件设计与优化，Andrew N.Sloss等著,北京航空航天大学出版社。