嵌入式系统
An Introduction to Embedded System

第七课、嵌入式实时多任务软件开发基础

课程大纲

- 嵌入式多任务软件开发过程简介
- 嵌入式多任务软件开发实例

嵌入式多任务软件开发过程

- 任务/中断划分
- 任务/中断优先级设计
- 任务/中断总体关联图、关联性分析（同步/互斥设计、优先级逆转分析）
- 任务可调度性分析
- 出错处理及恢复设计
- 任务代码详细设计

任务划分的目标

- 满足“实时性”指标
- 任务数目合理
- 满足操作系统裁剪要求
- 降低系统资源需求

任务划分的方法—设备依赖性任务的划分（1/2）

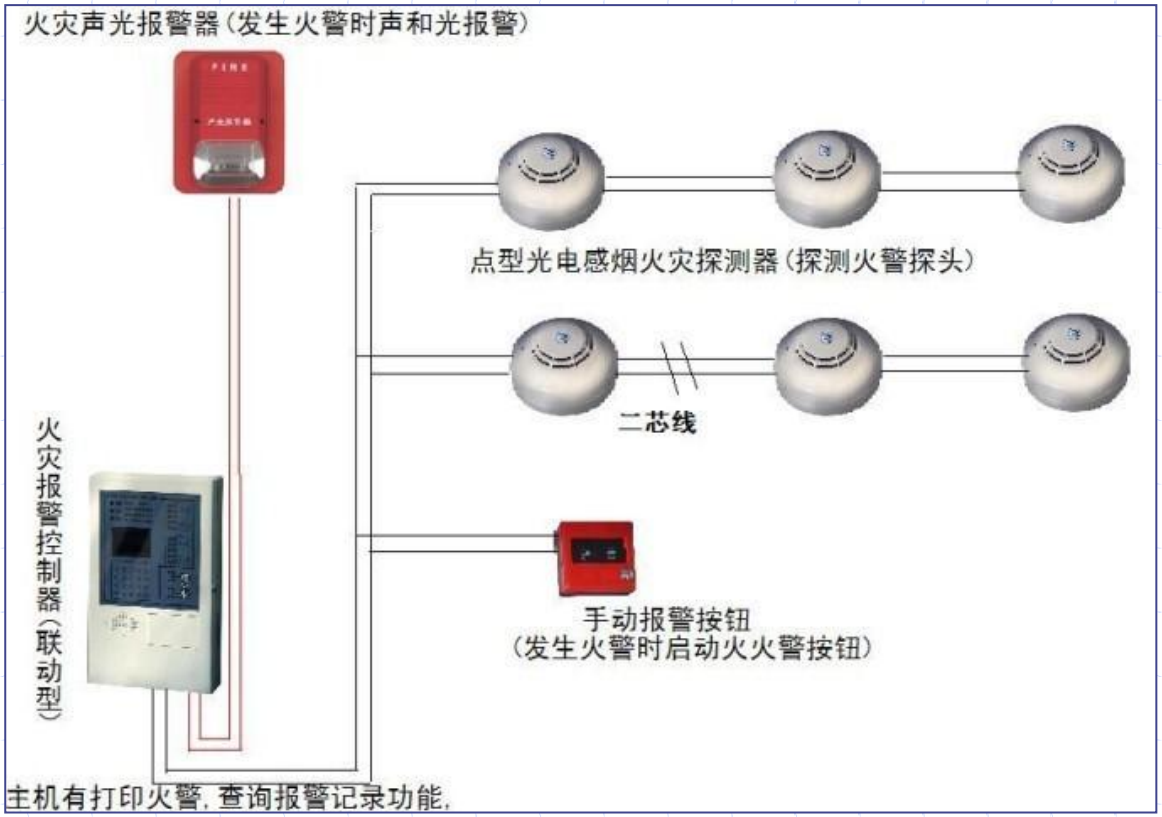
- 将系统中的各类输入、输出设备、控制系统封装成任务。
- 例如，针对手机系统的任务划分：
  - ✓ 键盘任务
  - ✓ 射频收发任务
  - ✓ 麦克风任务
  - ✓ 扬声器任务
  - ✓ 摄像头任务
  - ✓ 触摸屏显示任务
  - ✓ 有线通信任务
  - ✓ 电源管理任务

任务划分的方法—设备依赖性任务的划分（2/2）

- 例如，车载导航系统的任务划分：
  - ✓ GPS信号接收
  - ✓ 导航语音提示
  - ✓ 电源管理任务
  - ✓ 实时路况信息接收
  - ✓ 路径规划任务
  - ✓ 人机交互任务

任务划分的方法—关键任务的分离 (1/3)

- “关键”功能的任务指：系统中的这种功能若不能正常实现，将造成重大影响，因此，必须得到运行机会。
- 例如，超市的火警检测系统



任务划分的方法—关键任务的分离 (2/3)

- 超市的火警检测系统工作过程为：

  烟雾传感器—>自动报警—>启动喷淋灭火—>保存火警记录

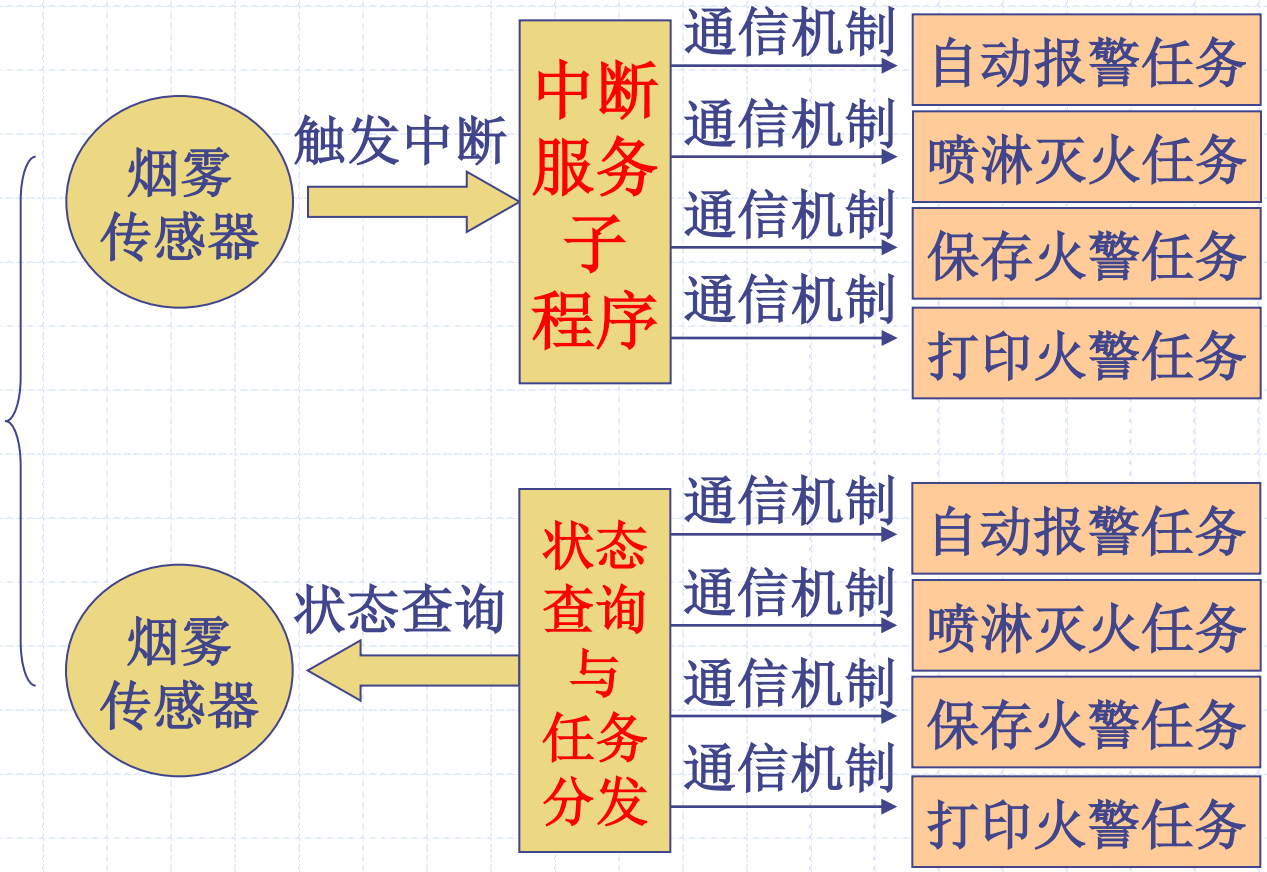
  —>打印火警记录

  其中的关键功能是：

  对烟雾传感器的检测

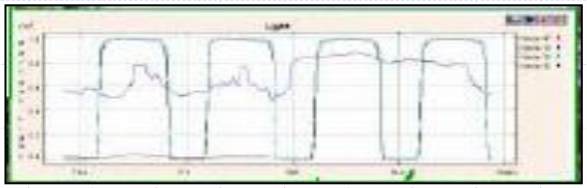
任务划分的方法—关键任务的分离 (3/3)

- 对烟雾传感器的检测，或者封装成中断服务程序（ISR），或者封装成足够高优先级的任务实现。

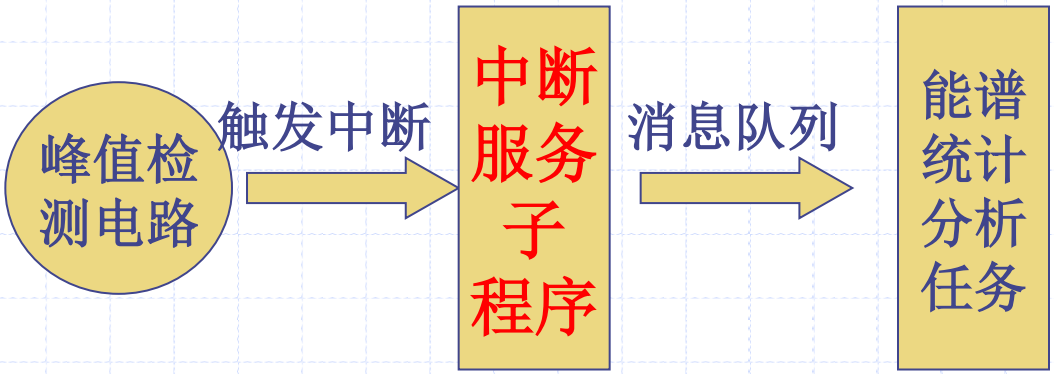


任务划分的方法—紧迫任务的分离

- “紧迫”功能的任务指：这种功能需在规定的时间内得到运行权，并在规定的时间内完成，即，具有严格的实时性。
- 例如，放射性测量中的能谱分析仪：将γ射线转换为电脉冲，对脉冲的密集程度与信号幅度进行分析。

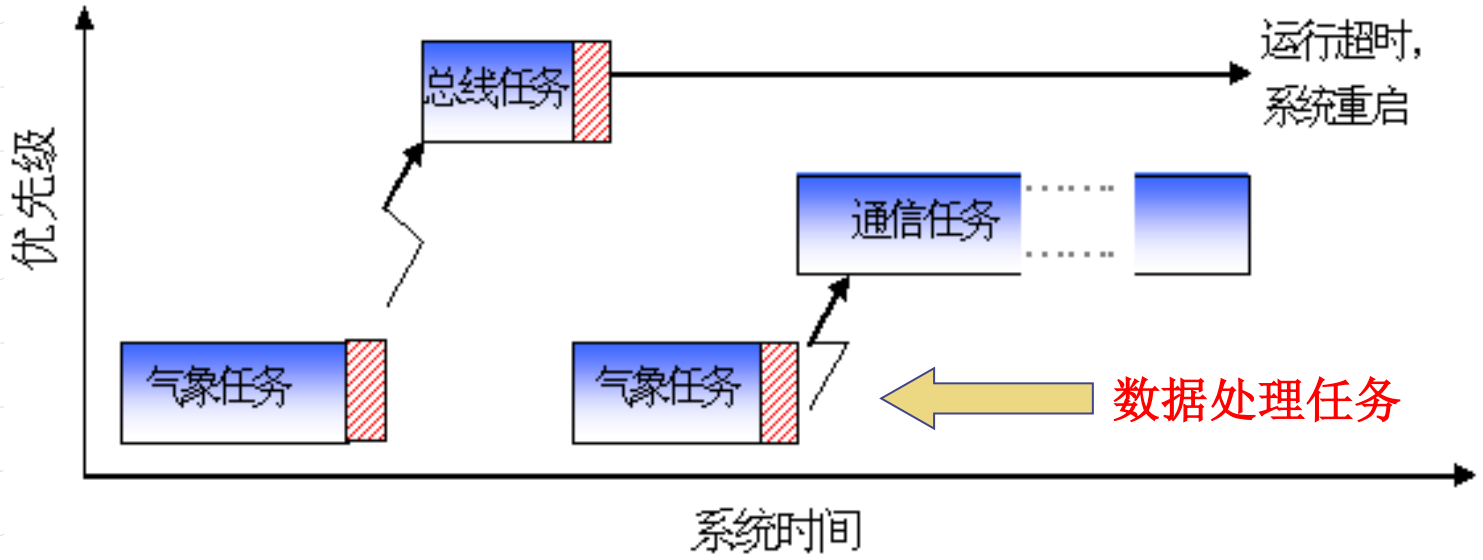


- 其中，能谱数据的采集是紧迫任务，希望不遗漏一个脉冲。



任务划分的方法—数据处理任务的分离

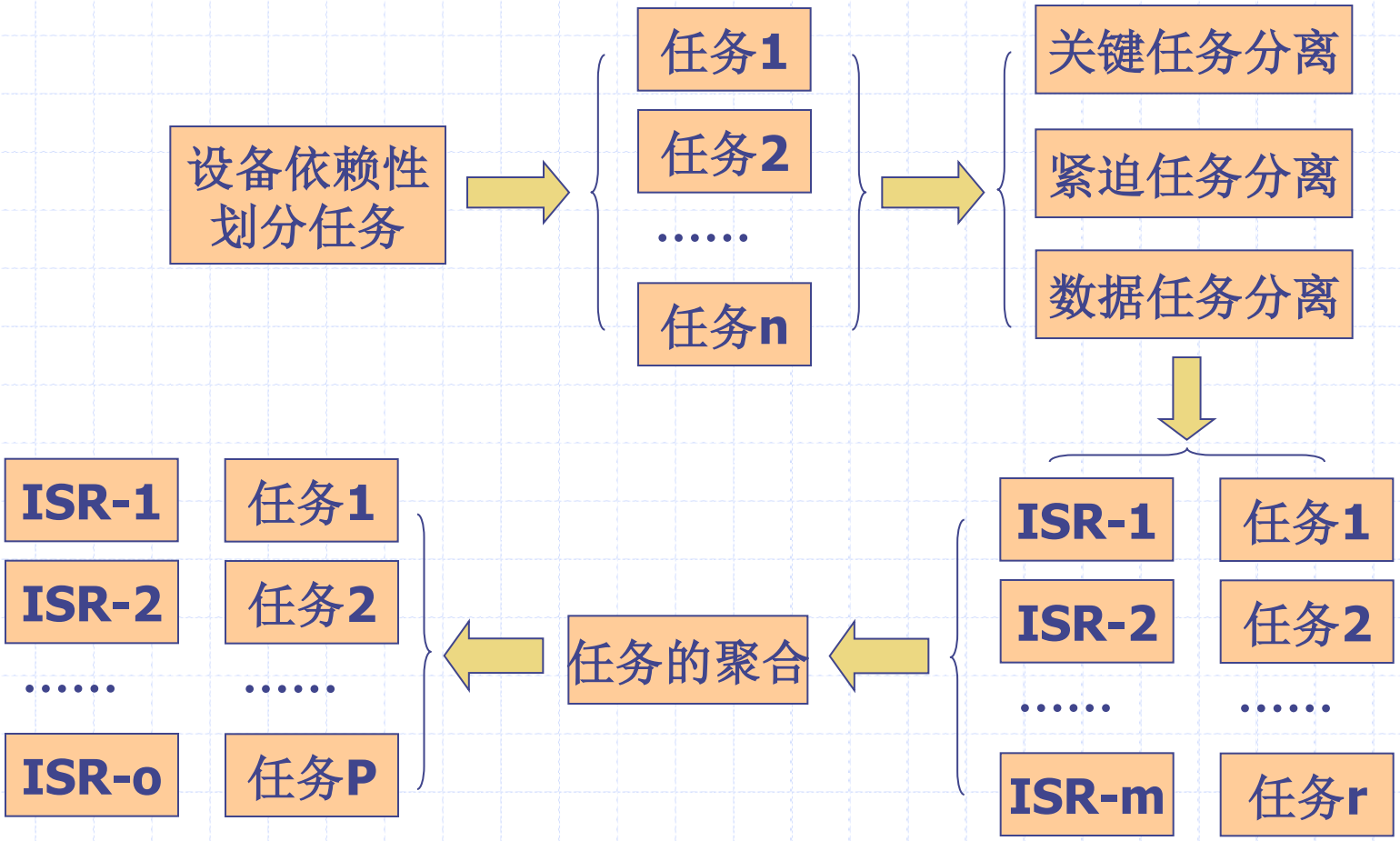
- 用户应用系统中消耗机时最多的是各类数据处理功能单元，应该将这些单元划分出来，分别封装成不同的任务。
- 这些任务的优先级可以安排得比较低。
- 例如：



任务划分的方法—任务的聚合

- 功能关联的任务聚合：将数据关联密切或时序衔接密切的功能单元组合为一个任务，减少数据通讯与任务同步。
- 触发条件相同的任务聚合：将相同事件触发的功能单元组合为一个任务，减少事件分发工作量。
- 运行周期相同的任务聚合。

任务划分方法一总结


设备依赖性
划分任务
任务1
任务2
……
任务n
关键任务分离
紧迫任务分离
数据任务分离
ISR-1
任务1
ISR-2
任务2
……
……
ISR-m
任务r
任务的聚合
ISR-1
任务1
ISR-2
任务2
……
……
ISR-o
任务P

任务/中断优先级设计

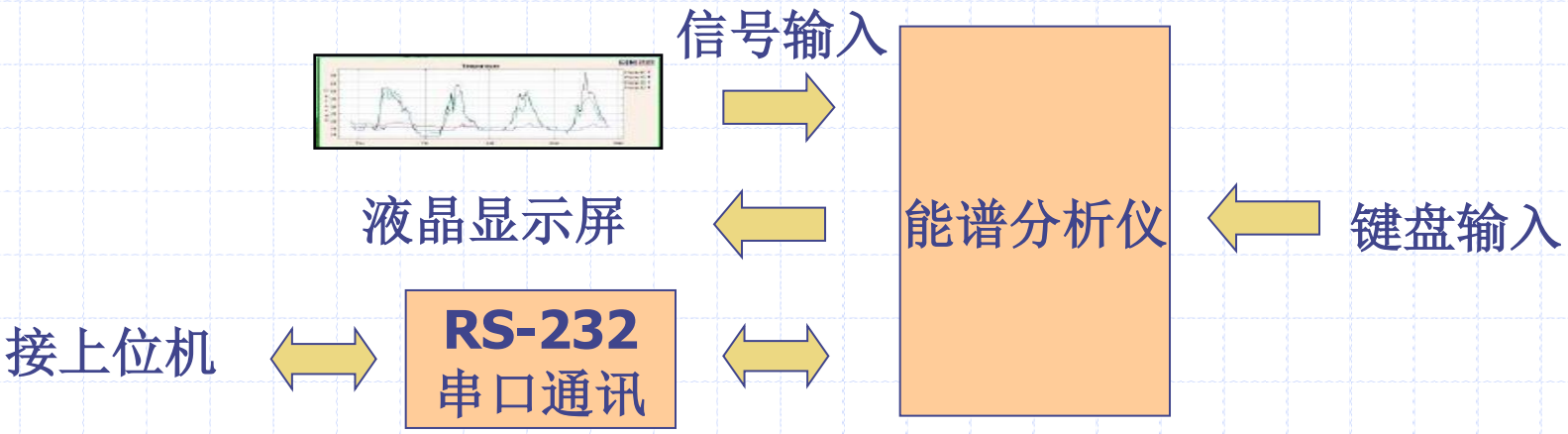
- 任务的优先级安排原则如下
  - 中断关联性
  - 关键性
  - 紧迫性
  - 频繁性
  - 快捷性
  - 传递性
- 中断的优先级安排原则如下
  - 关键性
  - 紧迫性
  - 频繁性
  - 快捷性

任务/中断关联性分析

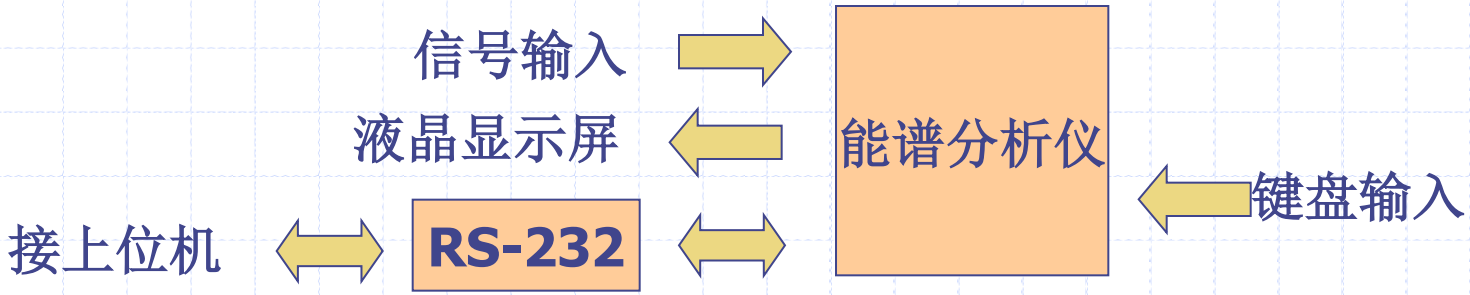
- 明确每一个任务与其他任务、中断服务程序间的关联关系，包括：行为同步关系、资源（数据）同步关系两类。
- 行为同步关系，主要包括：
  - ✓ 本任务（或ISR）的执行需要等待哪些任务（或ISR）发出的信号量或消息？
  - ✓ 本任务（或ISR）的执行可以向哪些任务（或ISR）发出信号量或消息？
- 资源同步关系，主要包括：
  - ✓ 本任务（或ISR）的执行需要等待哪些任务(或ISR)提供的数据？
  - ✓ 本任务（或ISR）的执行可以向哪些任务（或ISR）提供数据？

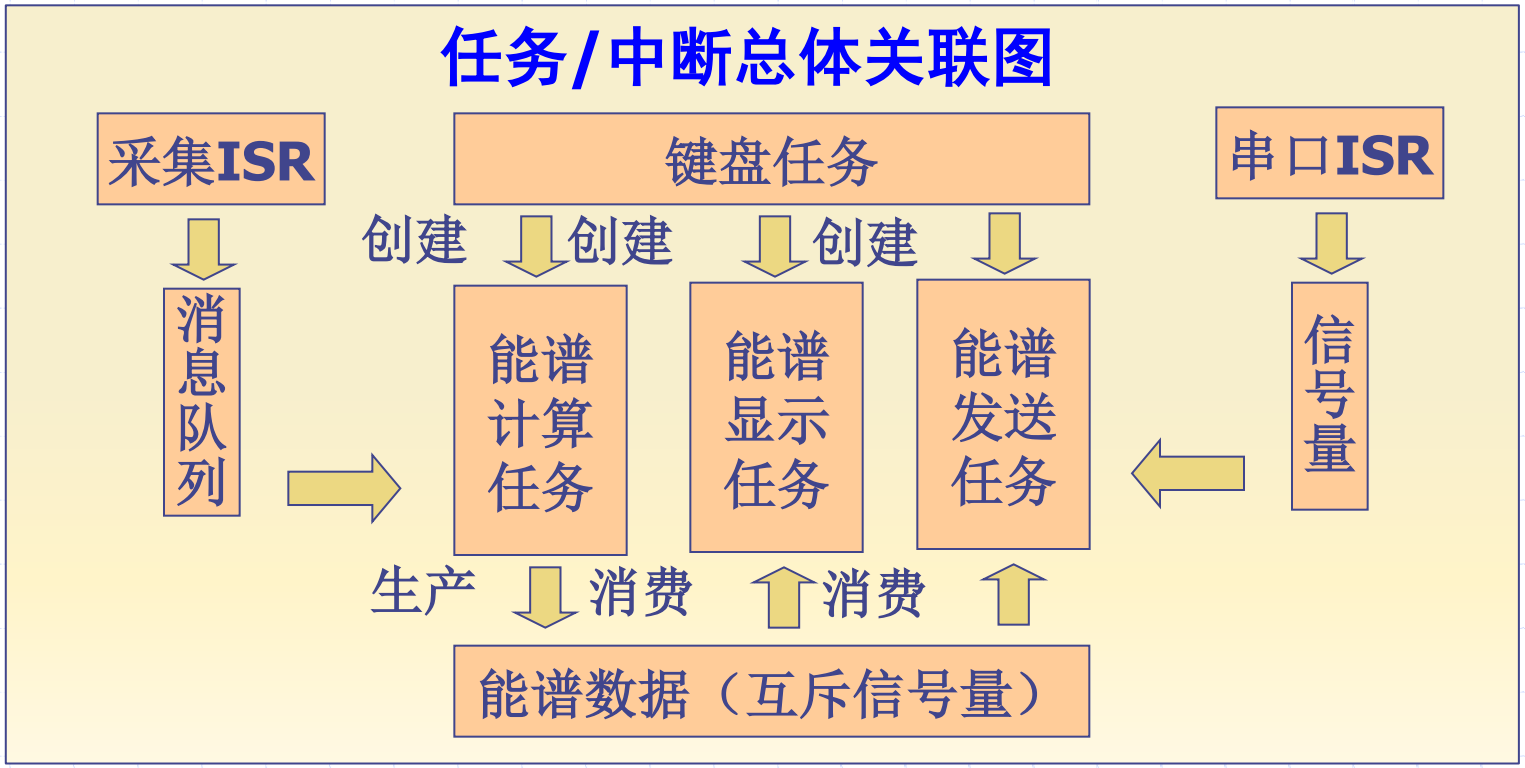
任务/中断总体关联图

- 当确定了任务划分后，需进一步把这些任务、中断服务程序之间的关联关系分析清楚，可以使用系统总体任务关联图来表示。
- 例如，放射性测量中的能谱分析仪原理框图如下：



任务/中断总体关联图—能谱分析仪





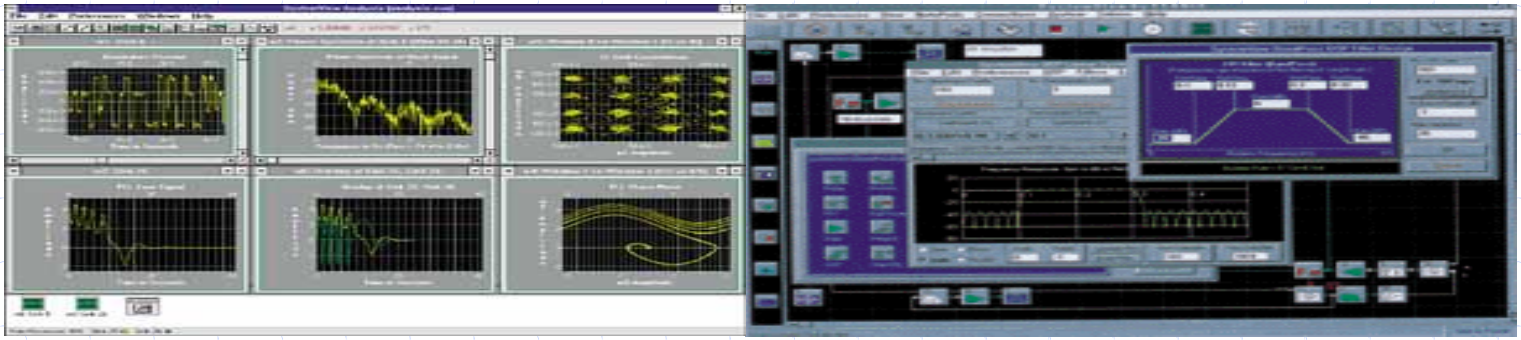
任务可调度性分析

- 任务划分及优先级安排后，需要进行“任务的可调度性分析”，以便确定是否可以在操作系统调度下正常运行。
- 任务的可调度性分析方法
  - 理论分析法：如RM算法中任务可调度性分析的一个充分条件

$$\sum_{i=1}^{n}\frac{C}{T}\le n\times(2^{\frac{1}{n}}-1)$$

CPU使用率上界

  - 实验仿真法：



多任务软件开发相关错误

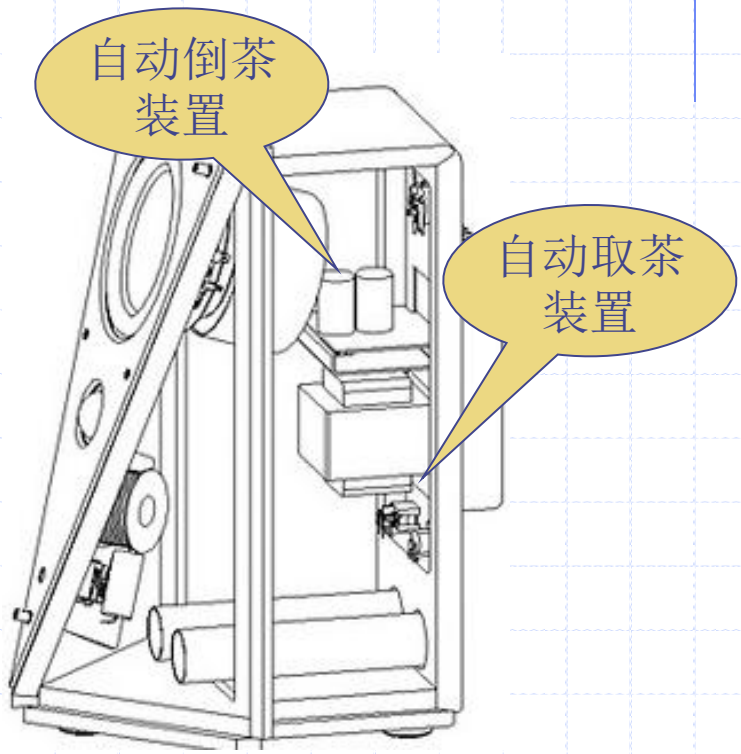
- 由于多任务开发中的缺陷所产生的问题，是嵌入式系统开发中最难解决的问题之一，直接影响到系统的可靠性、健壮性、执行效率和可维护性。
- 嵌入式多任务环境下可能出现的问题包括：
  - 划分问题：任务～中断
  - 优先级设置问题：任务～任务、任务～中断
  - 同步问题：中断～任务、任务～任务
  - 互斥问题：中断～任务、任务～任务
  - 通讯问题：中断～任务、任务～任务
  - 异常处理问题：出错与恢复、执行任务取消

课程大纲

- 嵌入式多任务软件开发过程简介
- 嵌入式多任务软件开发实例

多任务软件开发实例—自动售茶机

外形结构图



内部结构图

自动售茶机—任务划分

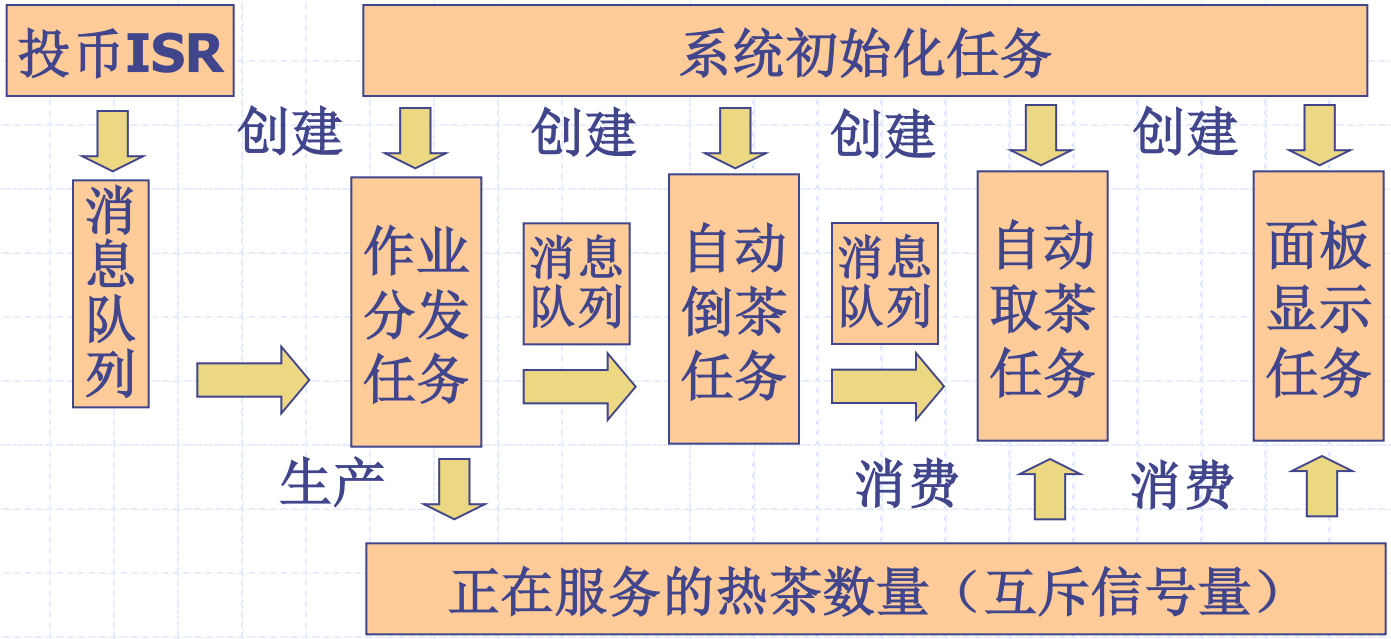
- 根据设备依赖原则，以及关键任务分离，将系统的任务划分如下：
  - ✓ 钱币输入系统：关键任务，ISR
  - ✓ 作业分发任务：重要任务，分发售茶任务
  - ✓ 显示屏：面板显示任务，一般任务
  - ✓ 自动倒茶装置：自动倒茶任务，一般任务
  - ✓ 自动取茶装置：自动取茶任务，一般任务

自动售茶机—任务优先级确定

- 系统的任务优先级由高至低，依次确定如下：
  - ✓ 投币输入请求：ISR
  - ✓ 作业分发任务：响应0.1s（依据：中断相关性、快捷性、传递性）
  - ✓ 面板显示任务：响应0.15s（依据：快捷性）
  - ✓ 自动倒茶任务：响应5s（依据：传递性）
  - ✓ 自动取茶任务：响应10s

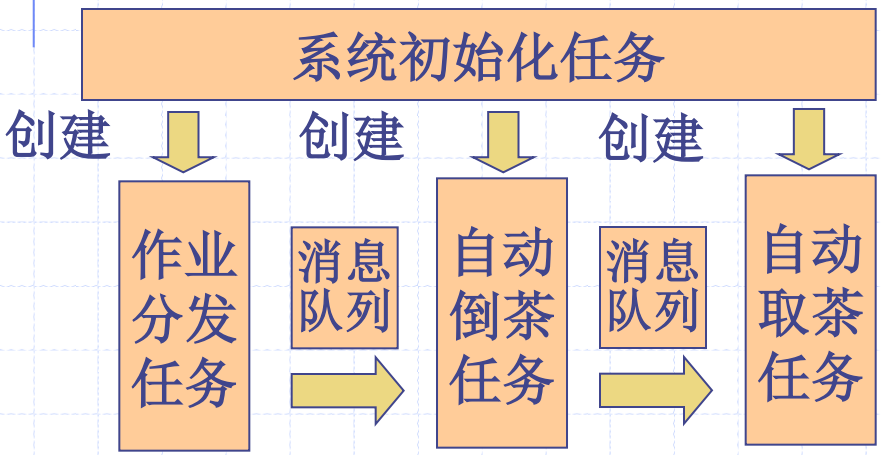
自动售茶机—任务/中断总体关联图

- 自动售茶机的任务/中断总体关联图如下所示：

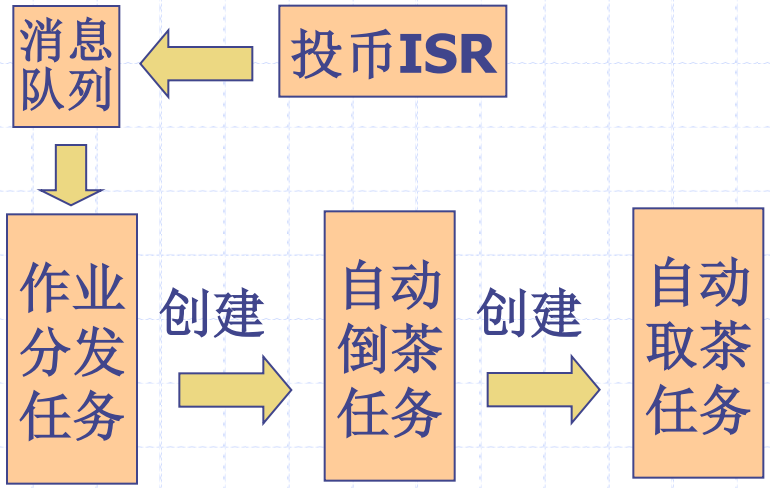


自动售茶机－多任务编程经验法则1

- 多任务创建静态化：所有任务最好能在初始化时创建完成，而不是动态创建，减小内存碎片；
- 优先级避免动态调整：任务优先级不要动态改变，有利于提高系统稳定性。



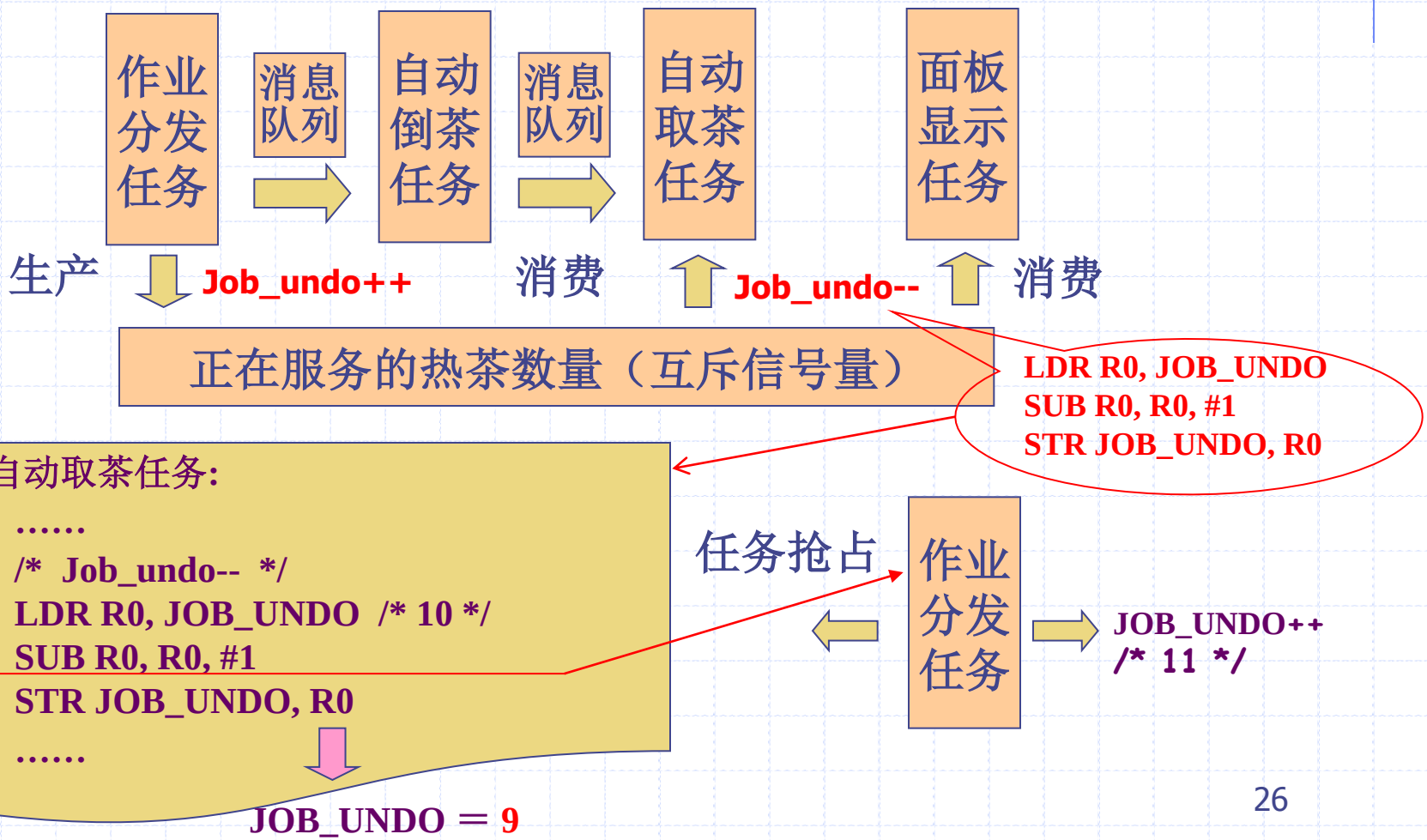
多任务静态创建



多任务动态创建

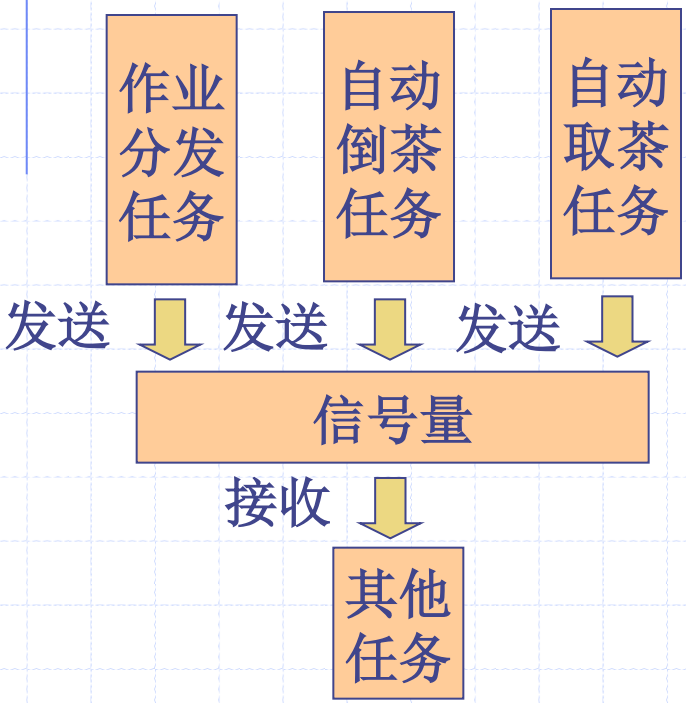
自动售茶机—多任务编程经验法则2

- 单条高级语言的操作不一定具有操作原子性，仍需进行互斥保护

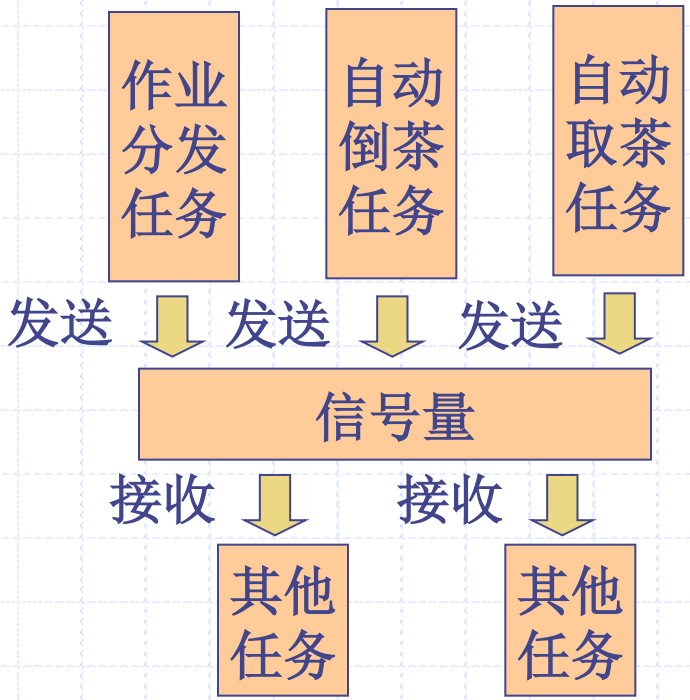


自动售茶机—多任务编程经验法则3

- 信号量任务同步的规范模式：采用信号量进行ISR～任务、任务～任务之间同步的规范模式包括：1：1、n：1、1：n；在实际应用中，n：m的信号量同步最好避免，可以采用事件机制。



n：1信号量同步



n：m信号量同步

自动售茶机－多任务编程经验法则4

- 基于RISC芯片的多任务共享变量、中断与任务共享变量时，必须使用volatile限定符，强制变量从内存中读取值。
- 例如，取茶设备必须在倒茶设备初始化完成后方能进行：

```
volatile int autotea_initflag = 0;
```

取茶设备初始化:

```
。。。
while (autotea_initflag != 1)
    taskSleep(100);
。。。
```

倒茶设备初始化:

```
。。。
if 倒茶设备初始化成功
  autotea_initflag = 1;
。。。
```

自动售茶机－多任务编程经验法则5

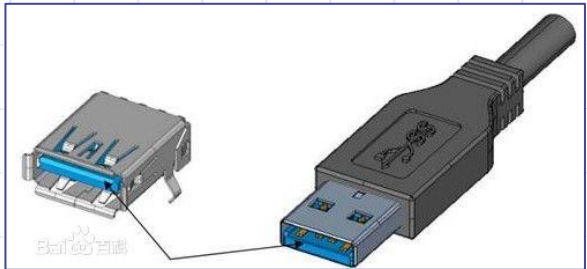
- 中断服务程序的处理时间要短：一般应小于100微秒，如果超过1毫秒，应将数据处理部分进行分离，或直接改为高优先级任务方式来执行。
- 例如：USB1.1协议标准的数据传输率为12Mbps，采用中断方式进行传输，每次传输数据块64byte，每秒中断数为：

总数据传输量（12×1024×1024）÷ 每次中断传输数据量（64×8）＝24576次/秒；

中断服务时间＝1 ÷ 24576＝40.69微妙

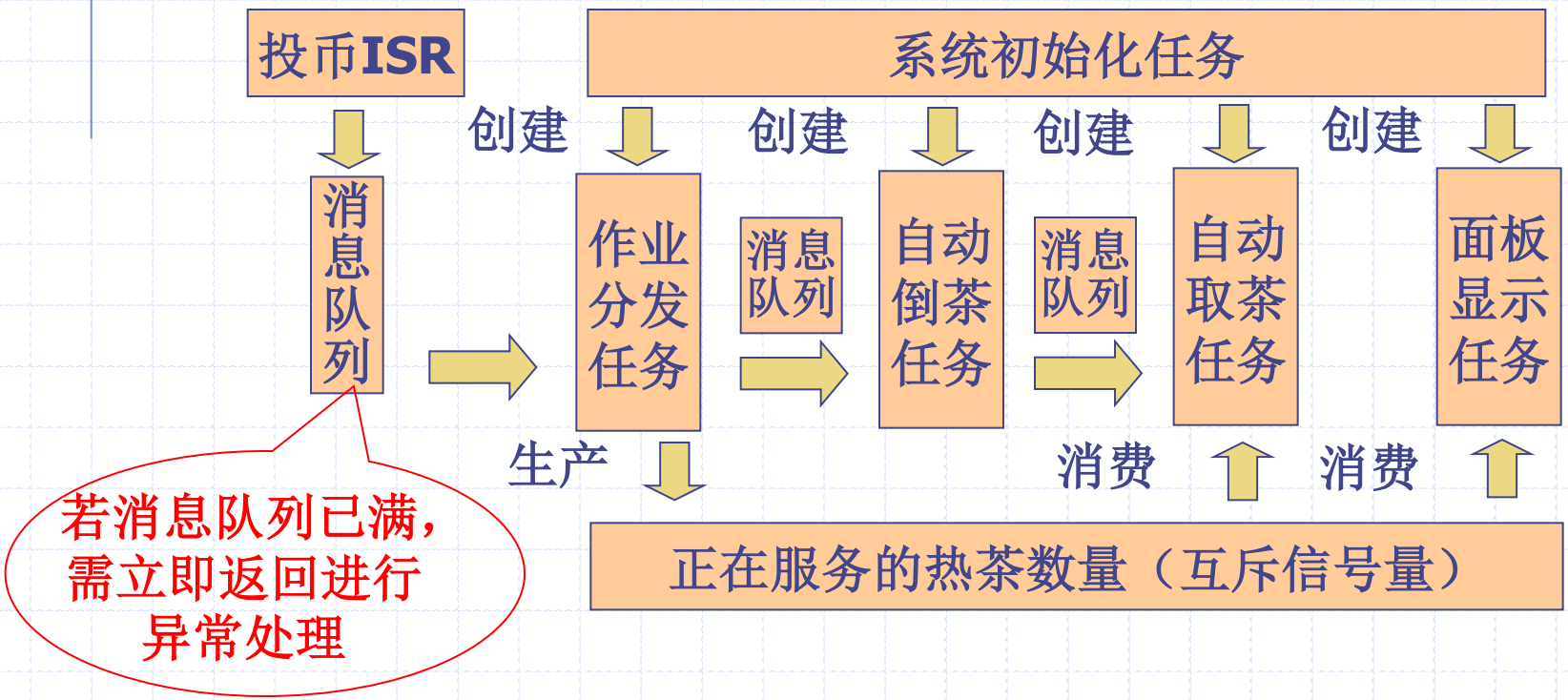
若ISR执行时间为1毫秒，它的传输速度仅为：

64×8×1000＝512000bps＝500Kbps



自动售茶机－多任务编程经验法则6(1/2)

- 中断服务程序不能被阻塞：ISR不能执行malloc/free、I/O库函数，或是接收信号量函数，在使用消息队列发送、接收函数时，一定要增加发送、接收失败后立即返回参数。



自动售茶机—多任务编程经验法则6(2/2)

- μC/OSII等操作系统，则对ISR不能执行的阻塞式系统调用，直接采取返回出错标志的处理方式。

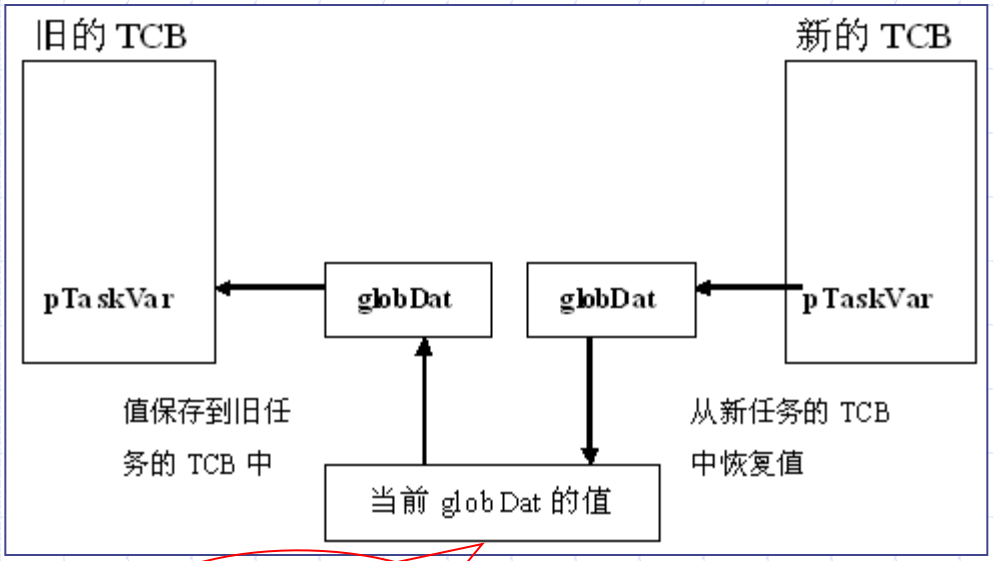
μC/OSII的消息队列发送代码

```
void  *OSQPend (OS_EVENT *pevent, INT16U timeout, INT8U *perr)
{
……
 if (pevent->OSEventType != OS_EVENT_TYPE_Q) {
     *perr = OS_ERR_EVENT_TYPE;
     return ((void *)0);
   }
   if (OSIntNesting > 0) {                /* See if called from ISR ...      */
     *perr = OS_ERR_PEND_ISR;             /* ... can't PEND from an ISR */
     return ((void *)0);
   }
   if (OSLockNesting > 0) {               /* See if called with scheduler locked ... */
     *perr = OS_ERR_PEND_LOCKED;   /* ... can't PEND when locked       */
     return ((void *)0);
   }
   OS_ENTER_CRITICAL();
   ……
   OS_EXIT_CRITICAL();
   ……
}
```

若在ISR中等待消息队列中消息，则返回错误

自动售茶机－多任务编程经验法则7(1/2)

- 多任务环境下的共享库需考虑可重入性：
  - 采用局部变量和函数参数来实现，是安全的；
  - 对全局变量和全局数据结构进行互斥保护，是安全的；
  - 采用任务变量。
- 多任务运行过程中，可能需要对每个任务有不同值的全局或静态变量。
- 嵌入式操作系统提供了任务变量机制，在任务的控制块TCB中增加一个变量，该变量的值，在任务切换时，被设置为任务的私有值。



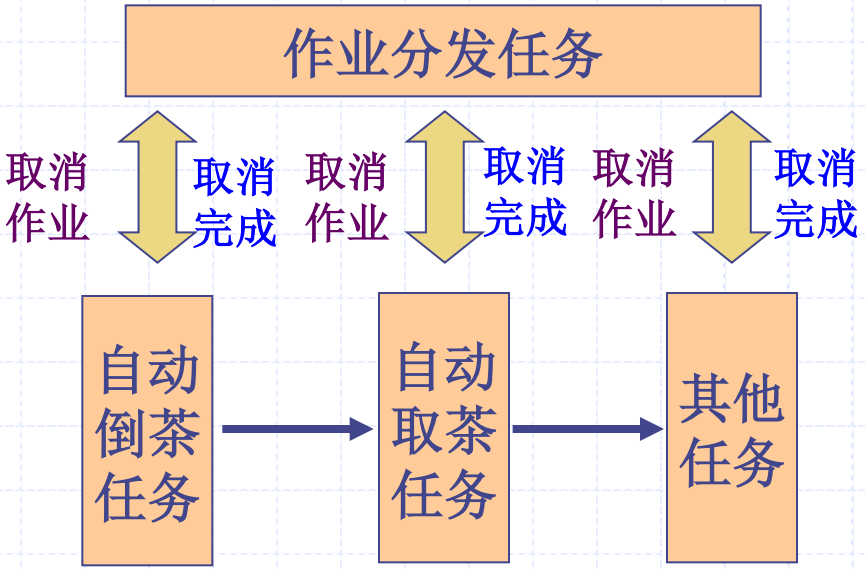
任务变量切换过程

自动售茶机－多任务编程经验法则7(2/2)

- 如果全局数据结构规模较小，而利用该数据结构进行数据处理的时间可能较长，可以先对全局数据结构进行复制后处理。
- 例如，需要实现一个基于字典编码的LZW压缩算法库函数，需要一个4096表项的数据字典。
  - 对数据字典作为全局数据结构进行保护，可能导致高优先级的压缩任务无法及时处理；
  - 可以对数据字典先进行复制，后进行压缩处理的方法进行。

自动售茶机－多任务编程经验法则8(1/2)

- 多任务环境下的出错及恢复处理：
  - 在单任务系统中出错处理和恢复比较容易，但多任务系统中的出错处理和恢复通常很困难。
  - 多任务系统中的出错处理，可以串行处理法，或并行处理法。
- 例如，自动售茶过程中用户多投了一些硬币，希望取消一些操作。

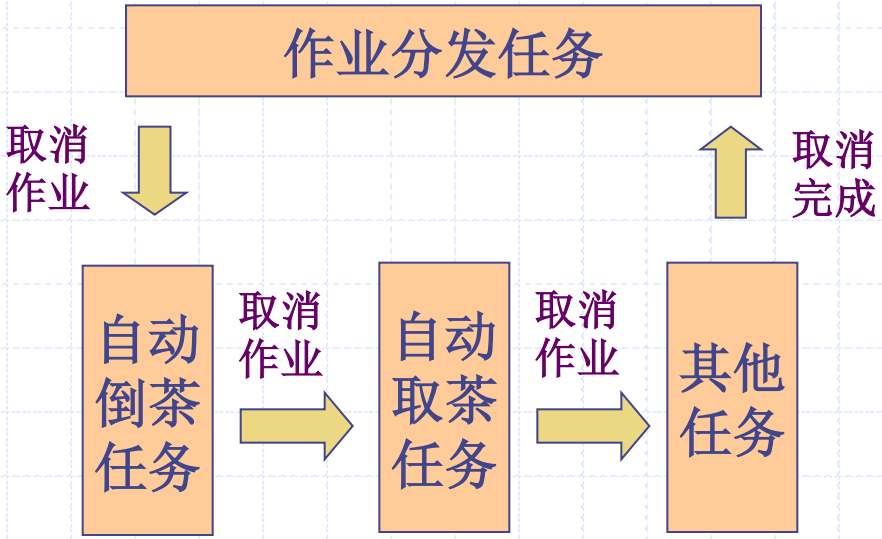


并行处理存在的问题：
残留作业的处理

取消作业后，各任务在恢复前，可能收到残余的任务。应丢弃残留作业。

自动售茶机—多任务编程经验法则8(2/2)

- 多任务环境下的出错及恢复处理：串行处理，可消除残留作业。



谢 谢！