嵌入实时操作系统的学习之道

-将理论付诸实践

何小庆 张爱华

2021年8月

1. 什么是嵌入式RTOS?
2. 嵌入式操作系统教学
3. 如何学习一种嵌入式RTOS
4. RTOS 优先级调度机制分析
5. RTOS 优先级调度实验

什么是RTOS?

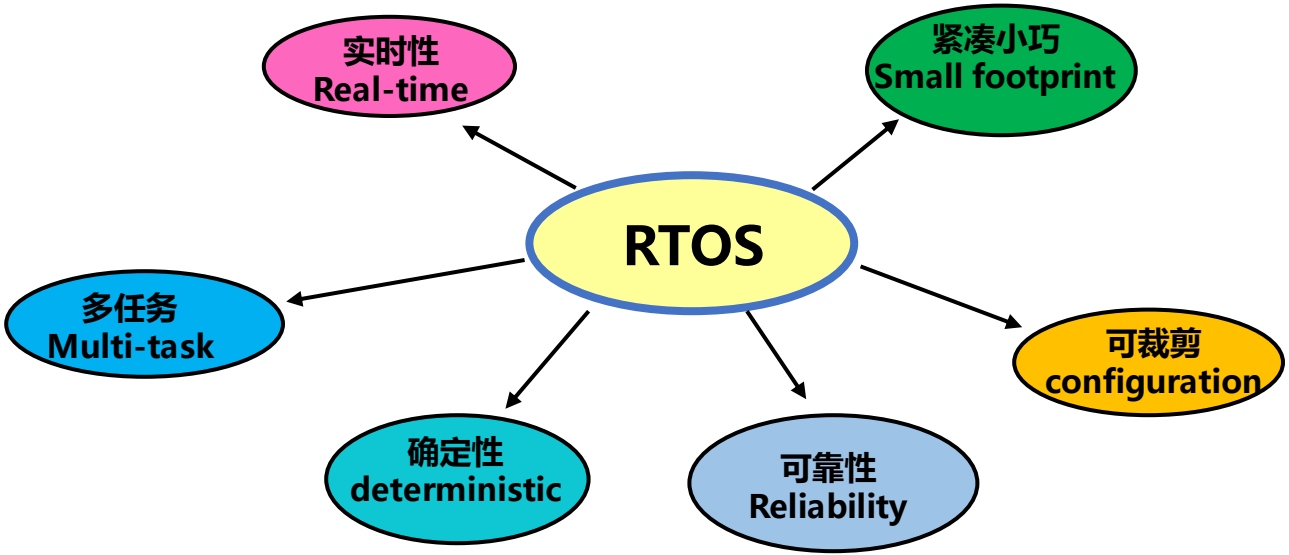
R(real) T(time) OS (Operating System)实时多任务操作系统

- IEEE的实时UNIX分委会认为RTOS应具备：
  1) 异步的事件响应。
  2) 确定的切换时间和中断延迟。
  3) 基于优先级的抢占式调度。
  4) 内存锁定。
  5) 同步机制。

- 产业界趋向认为：
  1) RTOS一种操作系统，属于嵌入式操作系统。
  2) RTOS 有三大特征：确定性，实时性和可靠性。
  3) 具有并行、异步和中断处理能力。
  4) 和其他嵌入式OS比较：RTOS 小巧，专用和可安全认证。
  5) RTOS 是物联网OS的重要基础。
  6) RTOS 内核有宏内核和微内核。

RTOS的特点

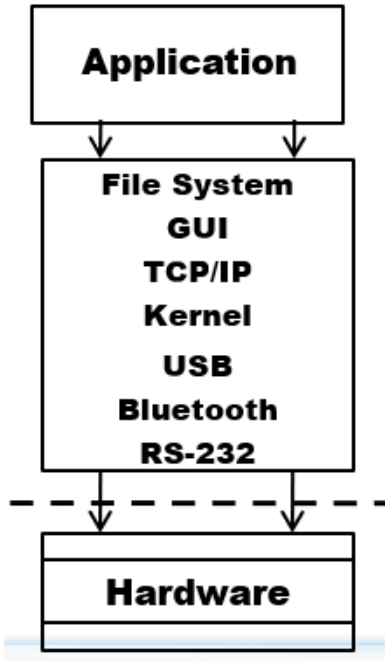
RTOS是一种操作系统，属于嵌入式操作系统，RTOS种类很多:有商业的、DIY和开源的，RTOS 是一种实时操作系统，而且是硬实时的操作系统



RTOS 与 RTOS Kernel

- **操作系统是一系列软件的集合，提供管理资源管理和应用代码服务的能力**
- **RTOS 除内核外包含了一系列的软件库（中间件）**

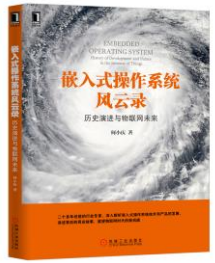
1) 操作系统与内核之间的界限并不明显，有时候还可以互换。
2) 一般讲：内核是操作系统的子集，它可以被视为将其他组件固定在一起的胶水。
3) FreeRTOS 和 µC/OS-III 是一个实时内核。
4) Vxwork 是一个嵌入式实时操作系统（RTOS）。



嵌入式实时操作系统发展历史

RTOS 名称	公司名称	网站	近况
VRTX	Ready System/Microtec		最早的商业RTOS
pSoS	ISI		在通信业知名的RTOS
OS-9	Microware		与MOT 紧密 历史悠久
SMX	Mico Digital	www.smxrtos.com	私人RTOS 企业
vxwork	Wind River	www.wrs.com	嵌入式RTOS 的常青树
Lynx OS	Lynuxwork	www.lynx.com	老牌的RTOS
QNX	QNX	www.qnx.com	汽车电子见长
CMX	CMX system	www.cmx.com	历史悠久,私人企业
Nucleus	ATI/Mentor	www.mentor.com	被Mentor-西门子 收购
ThreadX	Expresslogic	www.rtos.com	被微软收购-Azure RTOS
uc/OS	Micrium	www.micrium.com	被Silicon Lab 收购-开源
Integrity	Gree Hill	www.ghs.com	安全和军工
OSE	Enea	www.enea.com	通信业后起之秀
PikeOS	SYSGO AG	https://www.sysgo.com/	欧洲知名的RTOS
embOS	Segger	http://www.segger.com	工具见长

嵌入式OS始于和发展于RTOS，RTOS 有超过30年历史
全球兴旺的时候有几百家，中国也有自己RTOS



开源的RTOS

- RTEMS：实时多处理器系统，最早运用在美国国防系统。
- TOPPERS ： 日本开源的RTOS，创始人京都大学高田教授，现在专注在汽车电子。
- FreeRTOS ： 其专注在支持MCU ，开源模式和生态好，Amazon托管成为 IOT OS， 授权从GPL改成MIT。
- eCOS ： 基于GNU 的RTOS，含TCP/IP和文件系统，在消费电子产品有许多应用。
- Contiki：起源于无线传感网络的的RTOS ，有超低功耗管理和IPV6支持。
- µC/OS : 2020年 知名的RTOS µC/OS 变成开源软件。
- Zephyr：Linux基金会维护一个RTOS项目,Intel /NXP主导。
- Nuttx： POSIX API，无人机应用，小米Vela 的内核。

https://www.osrtos.com/

1. 什么是嵌入式RTOS?
2. 嵌入式操作系统教学
3. 如何学习一种嵌入式RTOS
4. RTOS 优先级调度机制分析
5. RTOS 优先级调度实验

“嵌入式系统”教学的现状

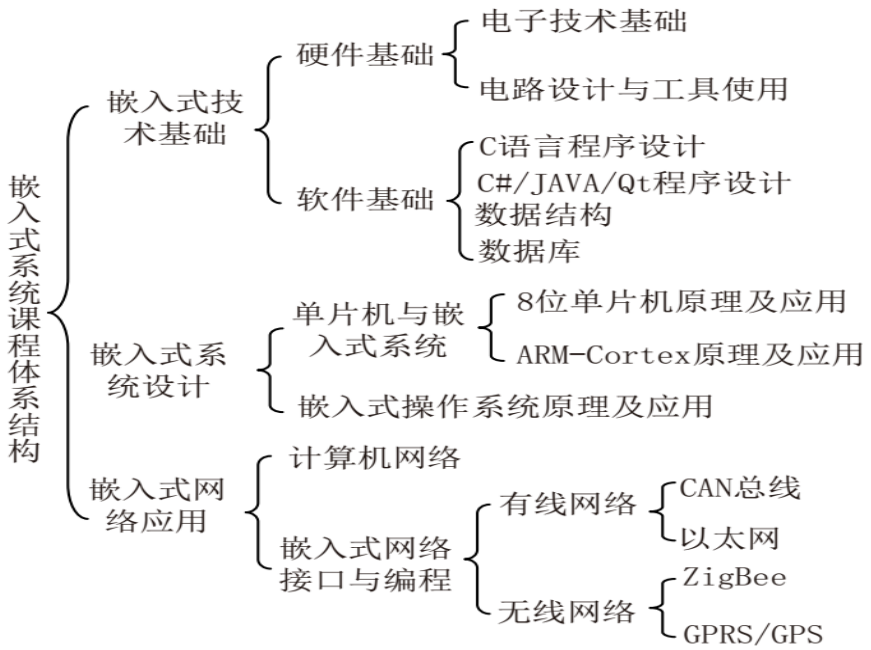
- 目前“嵌入式课程大致分成三类
  - **嵌入式系统硬件**

    -单片机、嵌入式系统体系结构，微机系统原理和SOPC/FPGA课程
  - **嵌入式操作系统**

    -Linux， uc/OS-2/RT-Thread
  - **嵌入式系统应用**

    -嵌入式网络、MP4/PDA和物联网应用场景（智能家居等IOT 课程）

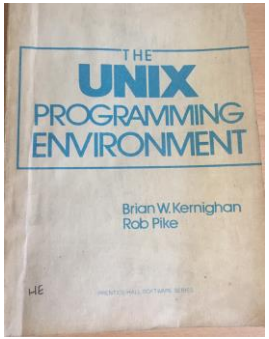
嵌入式系统课程教学素材最初来自国外。



来自：《物联网技术》

嵌入式课程目前内容跟不上技术与产业发展

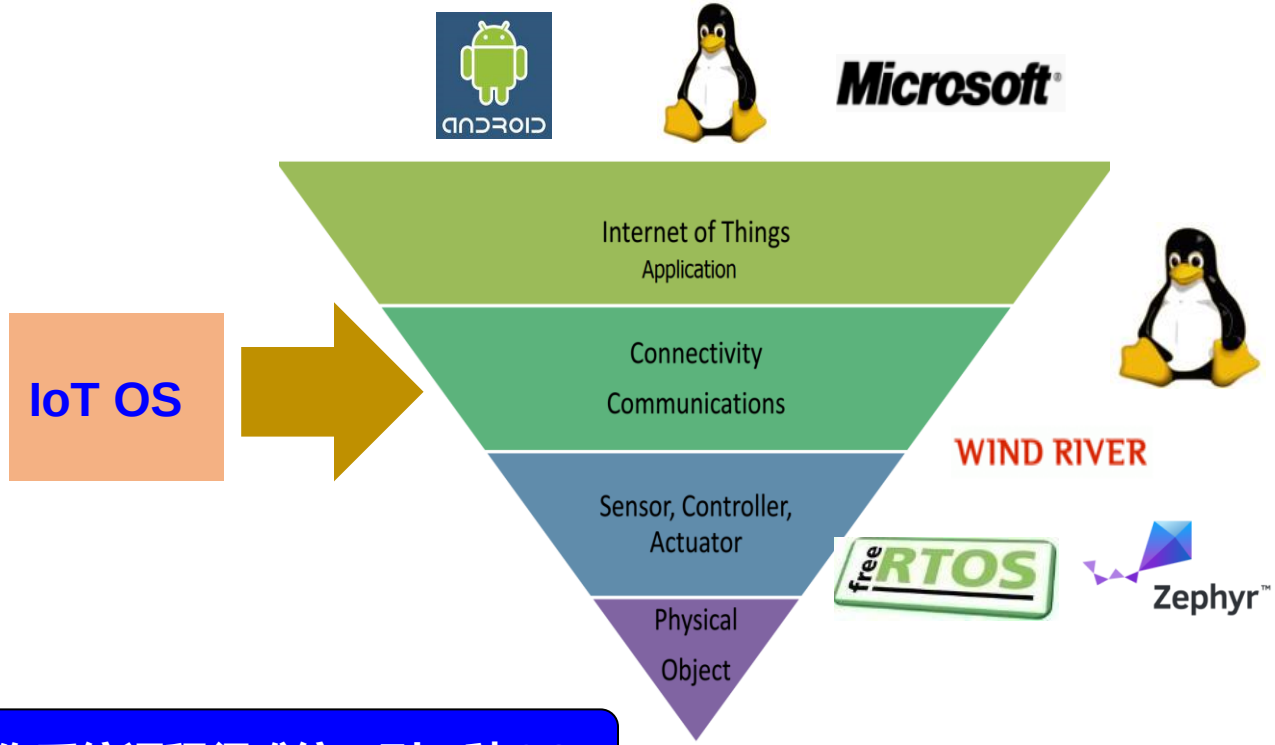
嵌入式操作系统课程回顾



- 操作系统课程由来已久
  - 一直以来高校以Linux 为主线
- 嵌入式操作系统课程实验
  - 以ARM Linux +QT为主线
- Android 开发应用
  - 智能手机应用开发
- 嵌入式系统开发课程
  - 部分以嵌入式Linux
  - 部分以RTOS(uC/OS和VXWORK)
- 单片机原理课程
  - 有一点RTOS（如uc/OS 和FreeRTOS 移植技术）

面向AIoT 嵌入式OS 课程如何变革?

一个OS可以支撑物联网系统的三层架构呢?



嵌入式操作系统课程很难统一到一种OS

嵌入式实时操作系统培训课程实践

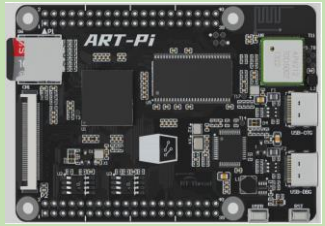
- 麦克泰和ST合作的RTOS培训课程
  - 基于FreeRTOS 初级课程
- 麦克泰开设FreeRTOS和 uC/OS高级课程
  - RTOS 原理和应用，嵌入式TCP/IP 和 IOT OS 简介
- 构建安全的嵌入式系统-从FreeRTOS到SAFERTOS
  - 以MPU技术基础，以FreeRTOS 和 SafeRTOS 为实战案例
- STM32教育联盟师资培训（TTT)
  - 精品课程“可穿戴系统设计与实现课程”

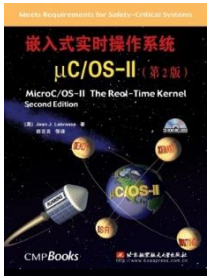
欢迎与老师在嵌入式OS课程建设上交流合作

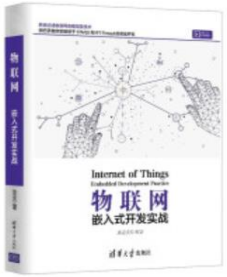
为嵌入式与物联网行业提供产品、教育和服务

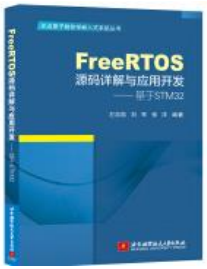
RTOS： µC/OS ,FreeRTOS 还是RT-Thread？

- uC/OS官方有非常好中英文文图书。
- µC/OS 以前是商业软件，现在是开源。
- FreeRTOS 是开源软件，FreeRTOS生态环境好，支持公司多。
- FreeRTOS 资料比较少，官方只有英文的PDF手册。
- RT-Thread 是国产开源RTOS，支持芯片种类多，国内生态环境好。









RTOS 原理非常接近，使用因人而异因事而异

为嵌入式与物联网行业提供产品、教育和服务

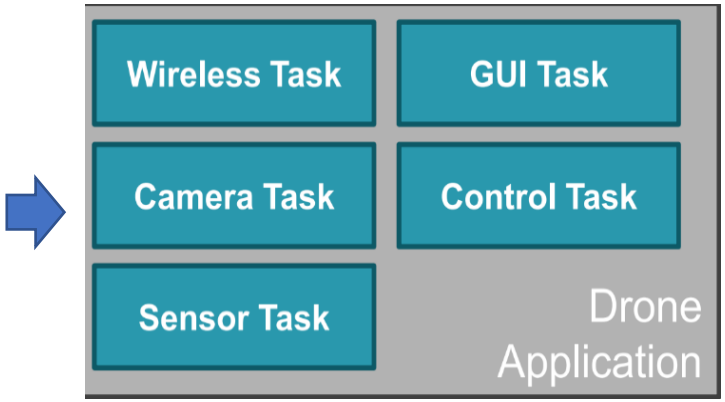
1. 什么是嵌入式RTOS?
2. 嵌入式操作系统教学
3. 如何学习一种嵌入式RTOS
4. RTOS 优先级调度机制分析
5. RTOS 优先级调度实验

RTOS 带来什么好处？ （1）

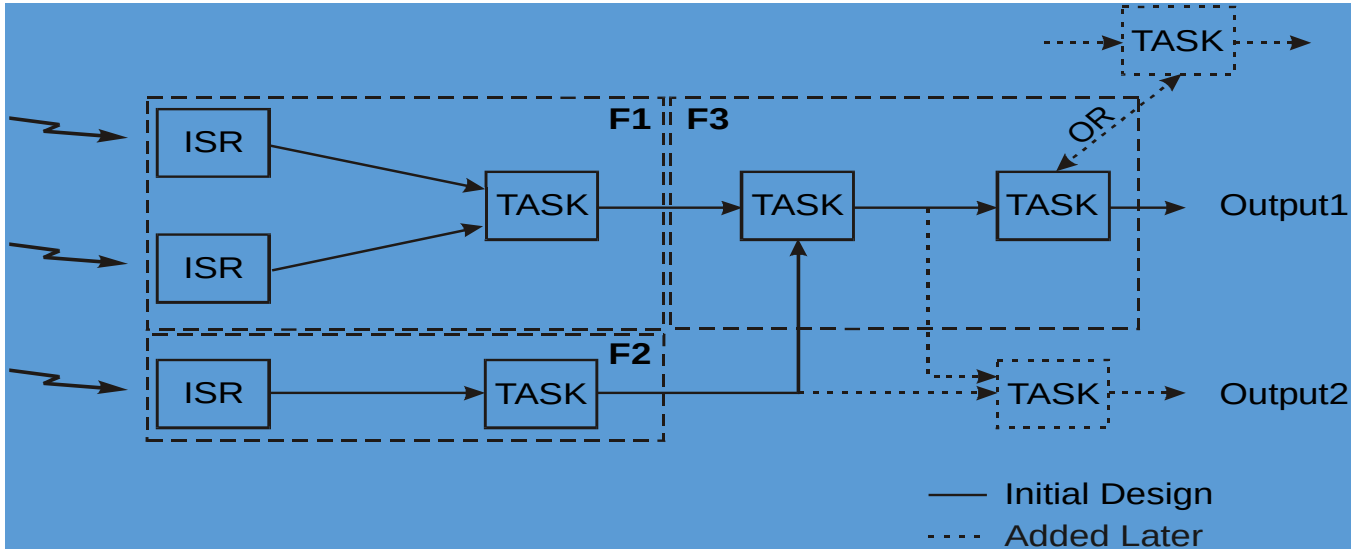
- RTOS 为你的应用提供服务。
  - 任务、队列、存储和时间管理。
- RTOS 方便增加软件组件
  - 为IoT 和HMI应用带来更多的便利。
- RTOS 是你开发应用的一个基础架构
  - 支持不同CPU 和 MCU
- 无论是商业还是开源RTOS bug 很少
  - 许多RTOS 通过安全认证
  - IEC-61508,ISO26262,DO-178B,
- RTOS 具备支持电源管理机制
  - 部分RTOS 内核也有低功耗调度支持。

RTOS 带来的好处？ (2)

- 支持并发处理（实时性）。
- 容易加入新的功能。
- 破解应用的复杂性。
- 方便系统更新和维护。



RTOS 的精髓 - 多任务系统



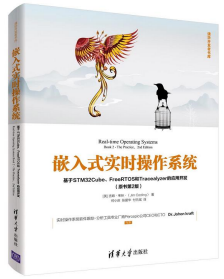
- 任务独立，基于优先级任务调度
- 任务间通信，异步处理
- F1，F2，F3三个功能模块接口清晰
- 易于加入任务

你是否需要一个RTOS？

- **你有实时性需求?**
  - *硬实时还是软实时，单处理器还是多处理器？*
- **你的设计中是否有独立的任务?**
  - *比如 UI, 控制循环和通信。*
- **你的系统中是否有高频度任务，它们会占用大量CPU时间?**
  - *比如不断更新GUI, 接受以太网帧和加密。*
- **你的项目的不同部分有多个开发者一起工作?**
  - *比如物联网设备。*
- **可移植和重用对于你是否重要?**
  - *公司有多个项目需要一个RTOS平台支撑。*
- **你开发的产品是否需要软件组件?**
  - *TCP/IP stack, USB stack, GUI, File System, Bluetooth 等。*
- **你的系统是否有足够的RAM支持多任务?**
  - *Flash 存储大小因素考虑不多，嵌入式系统一般有更多FLASH 而RAM 有时候比较少*

是否需要RTOS 也与生态环境有关

如何学习一种RTOS？



- 为什么要写这本书?
  - 当你想成为某个技术领域的专家时，你需要了解其理论知识。但是，[illegible]你想变得真正精通，那还远远不够-你还需要具有其“核心”的理解。我的意思是对这个领域有一种真实的感觉，我认为做到这一点的最佳方法是将**理论付诸实践，边做边学**。
  - 这在理论上似乎是个好主意，但实践却更具挑战性。①你需要一个方便实用的工具来完成工作。②对于许多自学者来说，工具一定不能很贵。③它们一定不难获得，使用和维护很方便。 这本书为你提供用于RTOS实验的低成本工具、软件和开发板的方法。
- 这本书的哲学
  - 本书的基本哲学是**“理解理论的最佳方法是将其付诸于实践”**。实验的目的是为你提供一条学习真正的商业工具的途径，实际工作从最简单的问题开始，然后逐步推进到更复杂的层面。如果你不熟悉即将开展的工作，请按照我们的顺序进行。

来自 Jim Cooling 的前言

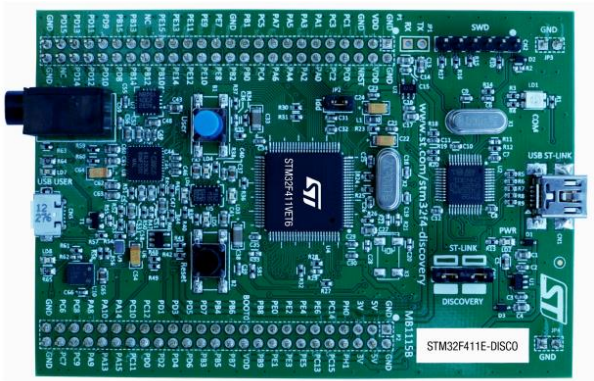
这本书的主要内容

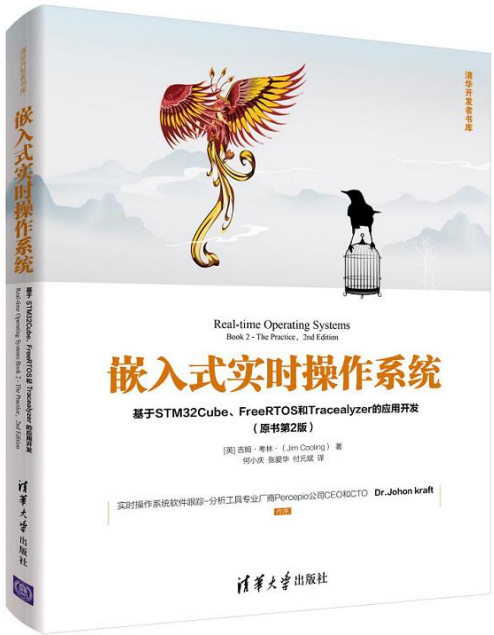
- 第一篇 应用代码开发
  - 第1章 开发流程及软硬件开发工具。
- 第二篇 内核基础实验
  - 第2章 多任务设计与实现基础
  - 第3章 共享资源使用
  - 第4章 任务交互实现
- 第三篇 使用Tracealyzer可视化软件行为
  - 第5章 Tracealyzer集成和配置指南
  - 第6章 Tracealyzer的基本特点和使用
  - 第7章 流模式操作介绍
  - 第8章 分析资源共享和任务间通信
- 第四篇 扩展你的设计知识、超越RTOS范围
  - 第9章 STM-Studio软件工具
  - 第10章 STM32F4通用定时器
  - 第11章 使用STM32F4看门狗定时器
  - 第12章 多任务设计中的通用任务故障检测技术
- 第五篇 结束语：展望未来
  - 第13章 自我改进指南
- 第六篇 帮助你自学的在线资料
  - 第14章 在线资料的参考指南

全书配合基础内核23个 可视工具实验11，定时器22个 共计56个

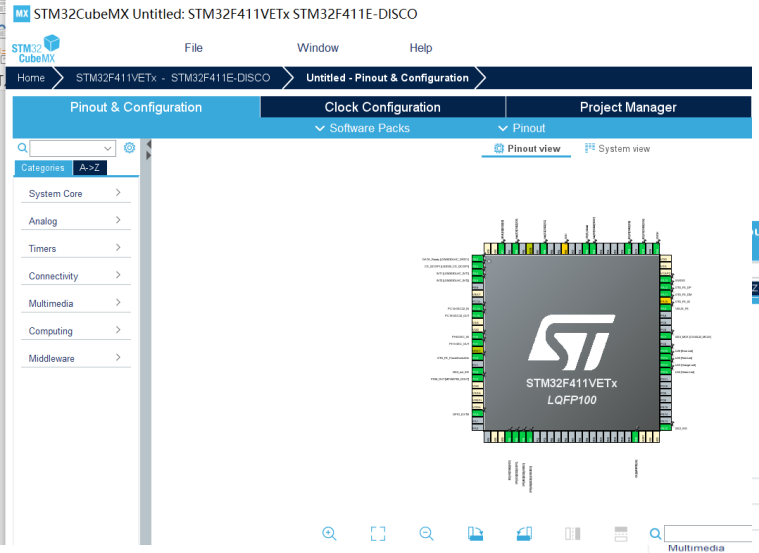
本书学习的工具和软件

- 基于STM32Cube、FreeRTOS和Tracealyzer学习RTOS原理
  - 软件环境
    - CubeMx v5.3.0
    - CubeIDE v1.0.2 （或者Keil /IAR)
    - Tracealyzer v4.3.4
    - STM-Studio
  - 硬件开发板
    - STM32F411EDISCOVERY （个别是在F407 板上验证）

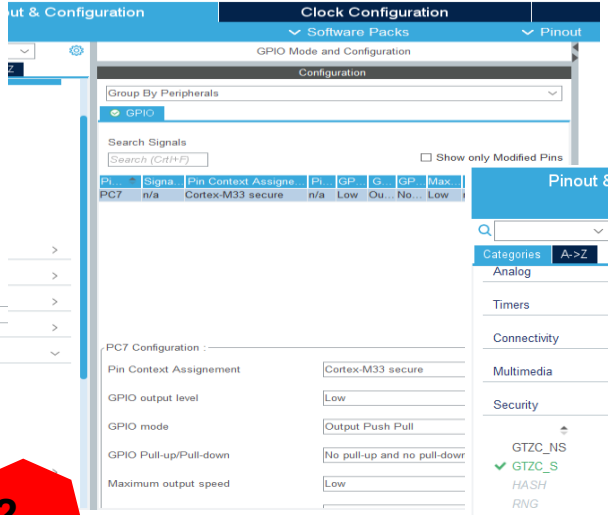




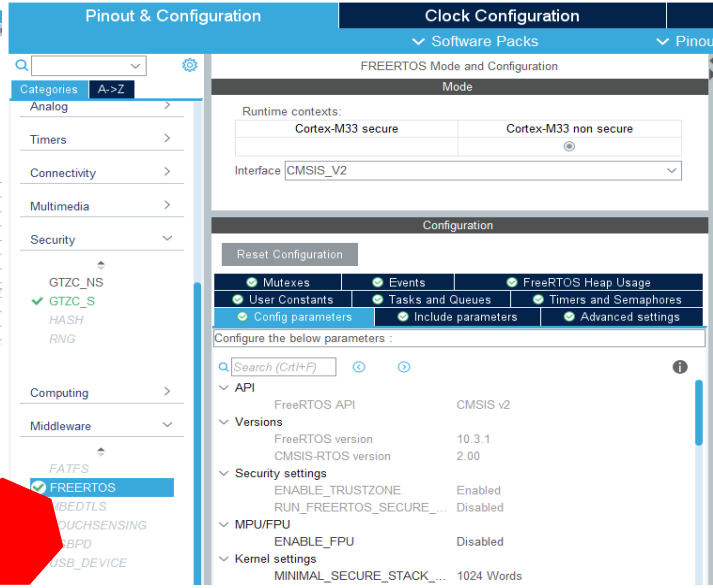
本书的实验开发流程（1）



1

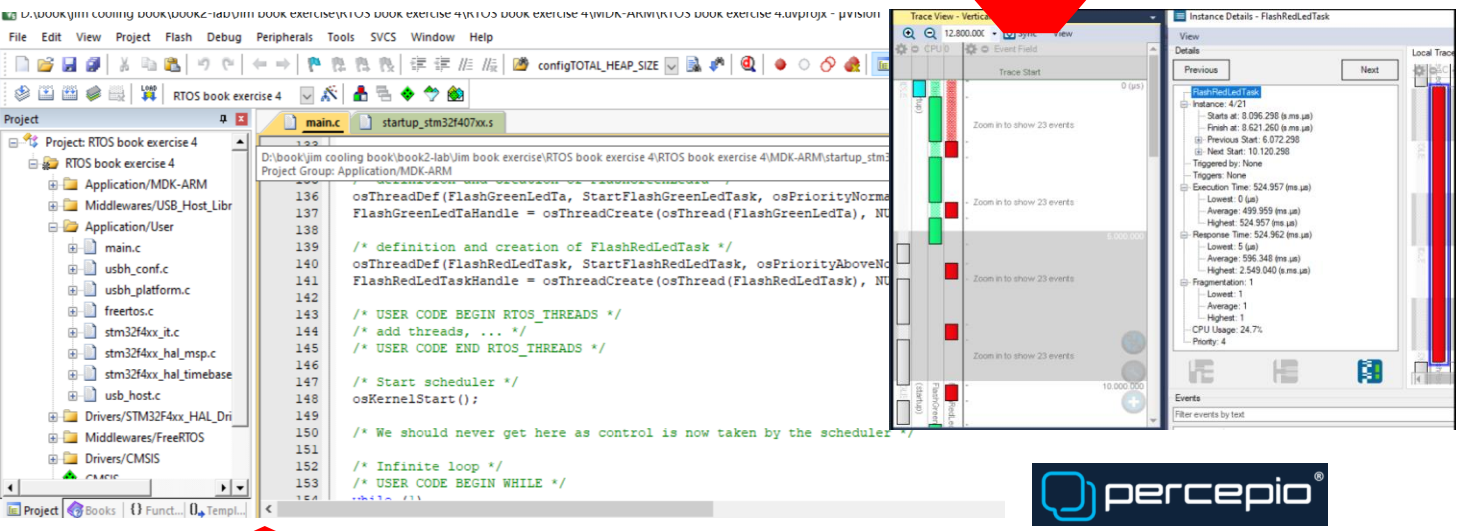


2



3

本书的实验开发流程(2)



4

7

percepio®

PC端

Keil MDK IAR EWARM 串口助手

USB

6

```
/* Create the thread(s) */
/* definition and creation of FlashGreenLedTa */
osThreadDef(FlashGreenLedTa, StartFlashGreenLedTask, osPriorityNormal, 0, 128);
FlashGreenLedTaHandle = osThreadCreate(osThread(FlashGreenLedTa), NULL);

/* definition and creation of FlashRedLedTask */
osThreadDef(FlashRedLedTask, StartFlashRedLedTask, osPriorityAboveNormal, 0, 128);
FlashRedLedTaskHandle = osThreadCreate(osThread(FlashRedLedTask), NULL);
```

freeRTOS

5

1. 什么是嵌入式RTOS?
2. 嵌入式操作系统教学
3. 如何学习一种嵌入式RTOS
4. RTOS 优先级调度机制分析
5. RTOS 优先级调度实验

RTOS 调度策略

- 什么是调度?
  - 调度解决的问题并不复杂：假设一个公司只有一辆卡车（CPU），但是有许多个司机（任务或者进程），任何时刻只有一个司机能使用卡车，然后每个司机只能接手一种类型的任务。在这样的背景下，货运经理如何能以“最好”的方式安排（“调度”）送货?
- 简单循环、定时循环和合作式调度
  - 协同调度- 由任务而不是调度器决定调度
  - 时间片轮询调度-定时循环调度
- 基于优先级调度策略
  - 抢占式调度
- 静态优先级和动态优先级
  - 编译时指定优先级，运行时可通过API更改
  - 运行时OS动态指定优先级

µC/OS-II® The Real-Time Kernel

µC/OS-III® The Real-Time Kernel

freeRTOS

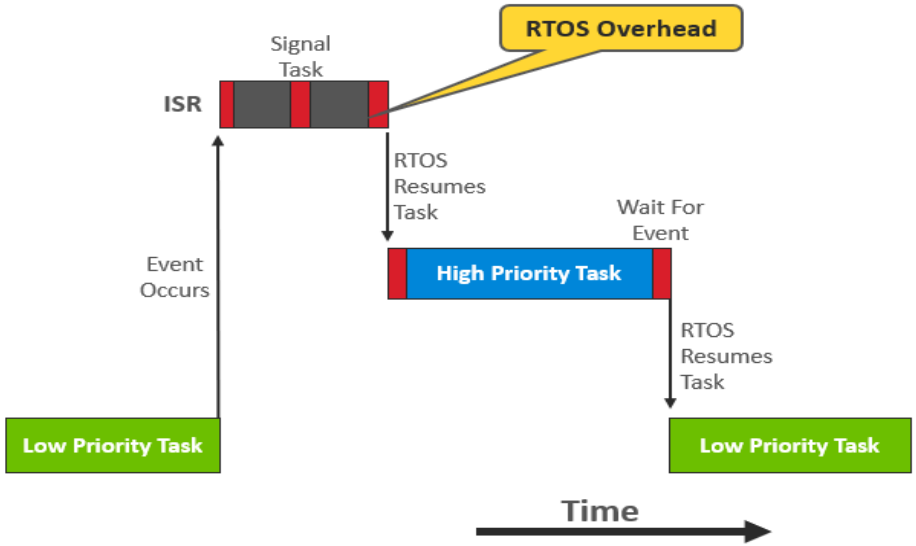
RTOS抢占调度策略

RTOS通过可抢占调度保证实时性

```
void Low_Prio_Task (void)
{
  Task initialization;
  while (1) {
    Setup to wait for event;
    Wait for event to occur;
    Perform task operation;
  }
}
```

```
void ISR (void)
{
  Entering ISR;
  Perform Work;
  Signal or Send Message to Task;
  Perform Work;   // Optional
  Leaving ISR;
}
```

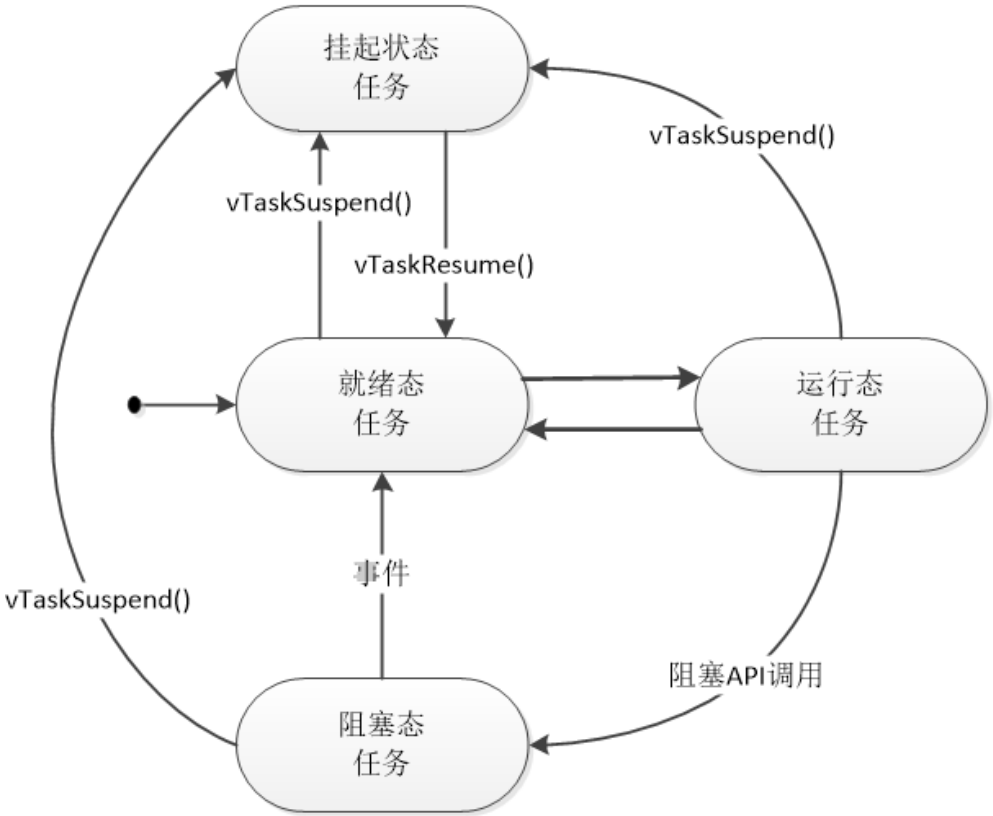
```
void High_Prio_Task (void)
{
  Task initialization;
  while (1) {
    Setup to wait for event;
    Wait for event to occur;
    Perform task operation;
  }
}
```



FreeRTOS任务状态

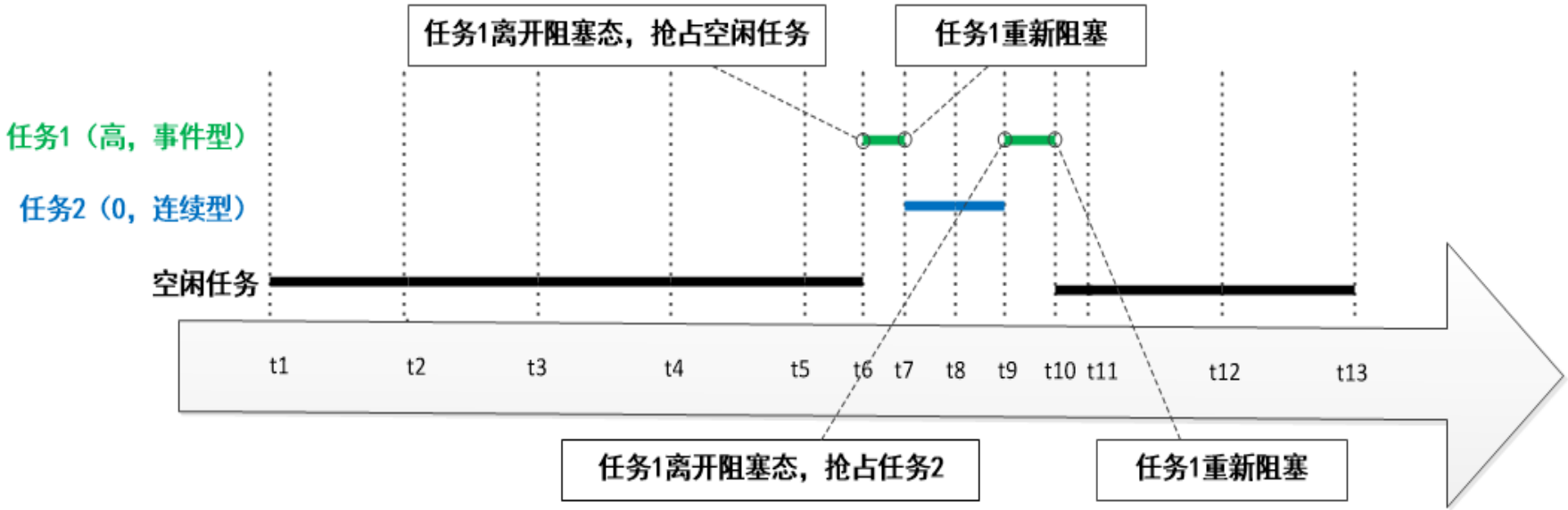
- 只有处于就绪态的任务可以分配CPU的使用权
- 每个任务都被分配了一个优先级，调度器使用优先级区分任务，

RTOS启动后，某个时刻执行哪个任务由调度器决定



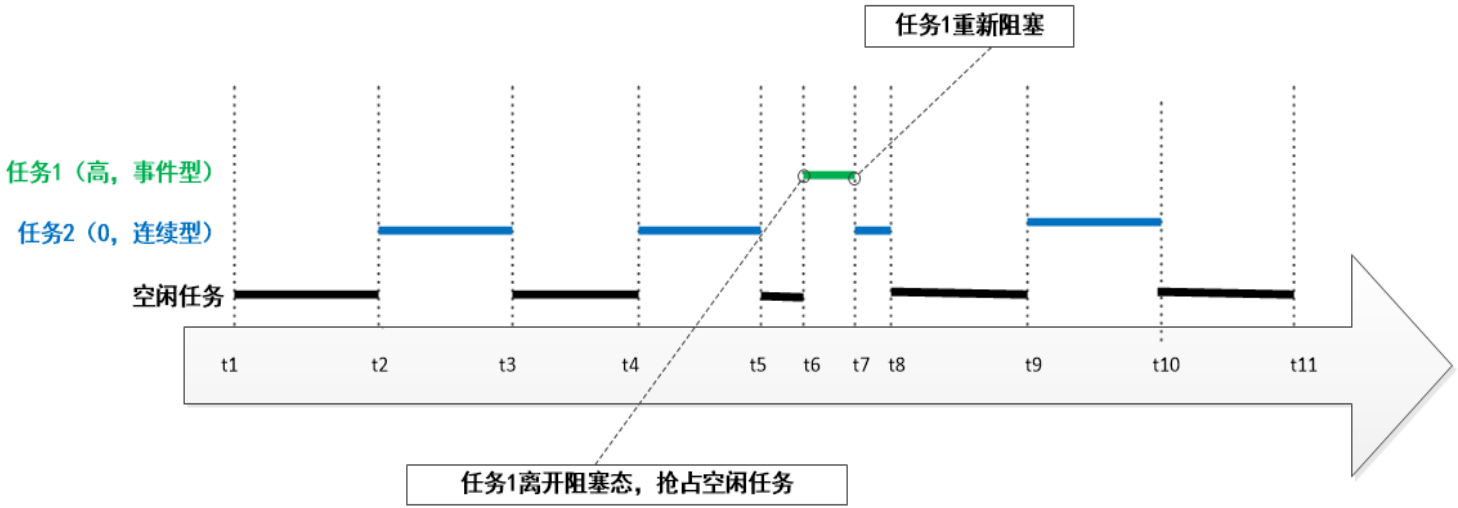
FreeRTOS任务抢占调度

- 就绪的优先级最高的任务获得CPU的使用权



FreeRTOS任务抢占+轮询调度

- 相同优先级任务之间按时间片轮转



1. 什么是嵌入式RTOS?
2. 嵌入式操作系统教学
3. 如何学习一种嵌入式RTOS
4. RTOS 优先级调度机制分析
5. RTOS 优先级调度实验

将RTOS理论付诸实践-实验范例简介

- 内核基础实验1-实验4（第2章）
  - 演示任务创建、优先级抢占调度
- 内核基础实验5-实验12（第3章）
  - 展示多任务系统中的共享资源争用问题，对系统性能的影响
  - 临界段代码保护机制
  - 系统安全提升
  - 优先级反转影响
- 内核基础实验13-实验23（第4章）
  - 任务交互机制
    - 同步-标志、事件标志、信号量
    - 数据共享-内存池、队列、邮箱
  - 按键中断服务
    - 为何需要快速中断处理
    - 使用可延期服务器减少ISR影响

本书涉及主题

- 开发流程及软硬件开发工具
- 多任务设计与实现基础
- 共享资源使用
- 任务交互实现
- Tracealyzer集成和配置指南
- Tracealyzer的基本特点和使用
- 流模式操作介绍
- 分析资源共享和任务间通信
- STM Studio软件工具
- STM32F4通用定时器
- 使用STM32F4看门狗定时器
- 多任务设计中的通用任务故障检测技术

理解抢占任务行为

- 嵌入式系统设计的基本要素是充分理解任务运行时行为
- 如何直观了解优先级抢占调度策略时的任务行为?
  - 内核基础实验4（P59）
    - 通过定性演示，帮你“感觉”使用抢占调度策略时的任务行为

2.5.1 背景介绍

嵌入式多任务处理系统设计的一个基本要素是充分理解任务运行时的行为，特别是领会优先级抢占调度如何及为什么会极大地影响性能。因为，如果不理解该问题，可能会导致设计因为无法预测的问题而终止运行。

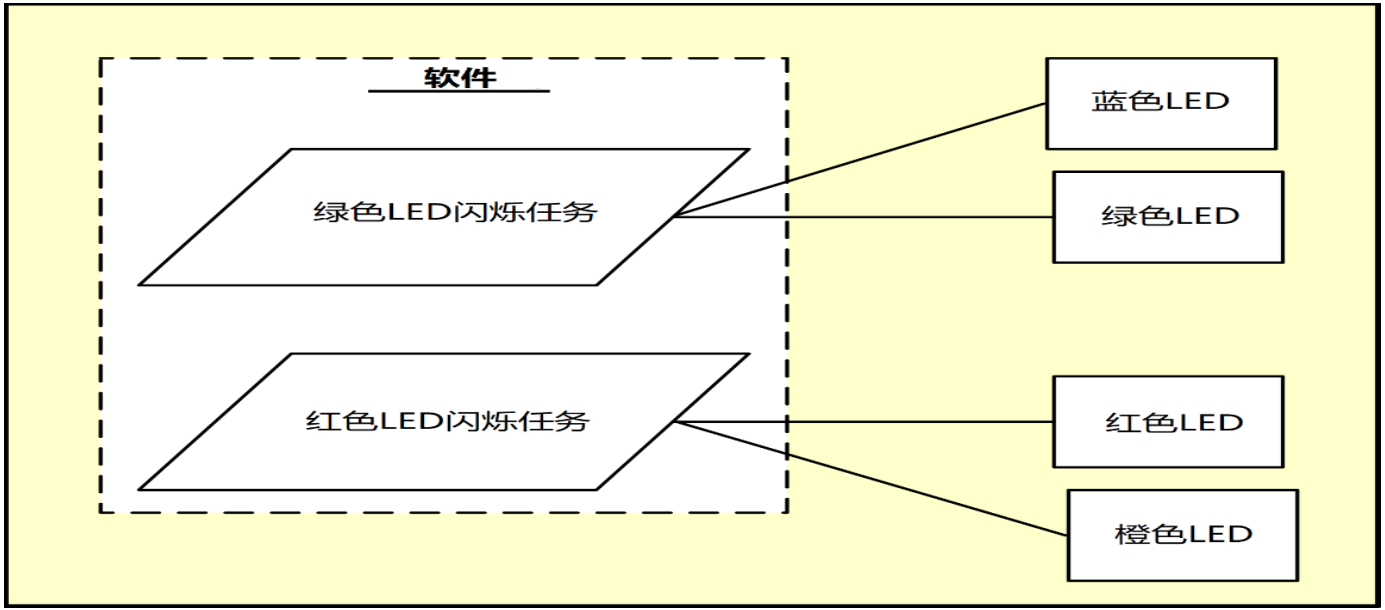
本实验的目的是引导你完成多个场景实现，每种实现都旨在演示任务之间相互干扰所产生的各种效果。注意：实验并不是要对这些影响进行定量演示（你需要合适的运行时分析工具实现该目的）。相反，它是定性演示，帮你“感觉”使用抢占调度策略时的任务行为。该工作基于两个任务的设计模型实现，每个任务控制两个 LED 灯，通过观察运行时灯的状态变化可以确定整个系统行为。

摘自：嵌入式实时多任务操作系统 清华大学出版社 P59

内核实验4-任务设计

提供核心任务实现c代码
http://hexiaoqing.net/publications/

任务模型



内核实验4基于两个任务的设计模型，每个任务控制两个LED灯，通过观察运行时灯的状态变化可以确定整个系统行为

内核实验4-任务实现

FlashGreenLedTask任务

周期时间：10秒

任务执行时间： 4秒

FlashGreenLedTask任务结构

无限循环开始

点亮蓝色LED灯

模拟任务执行-绿色LED灯闪烁4秒

熄灭绿色LED灯

熄灭蓝色LED灯

任务挂起6秒（使用osDelay函数）

无限循环结束

FlashRedLedTask任务

周期时间：2秒

任务执行时间：0.5秒

FlashRedLedTask任务结构

无限循环开始

点亮橙色LED灯

模拟任务执行-红色LED灯闪烁0.5秒

熄灭红色LED灯

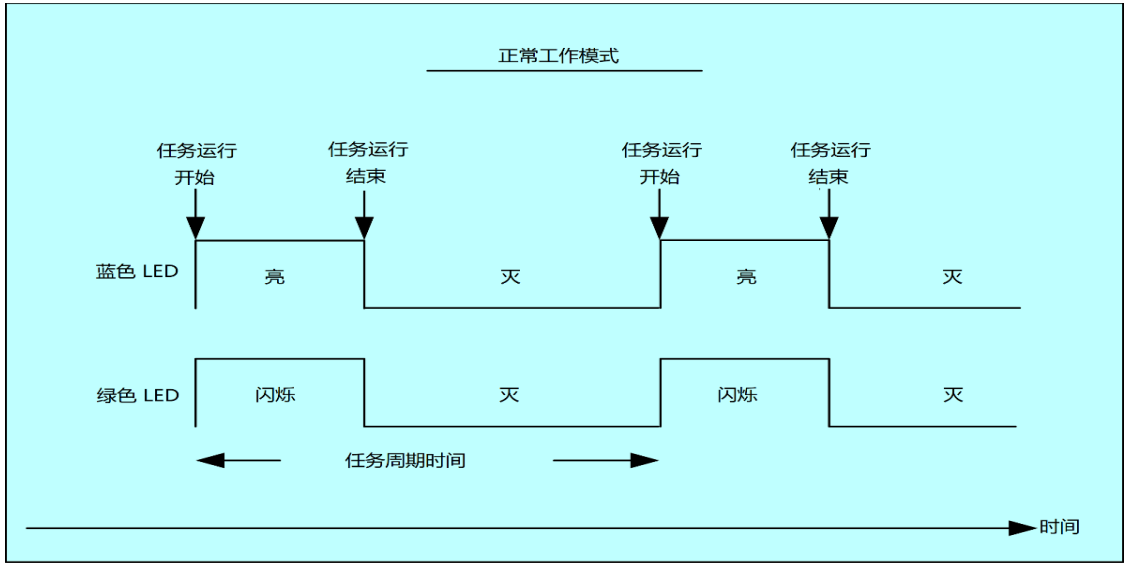
熄灭橙色LED灯

任务挂起1.5秒（使用osDelay函数）

无限循环结束

内核实验4-任务行为时序（1）

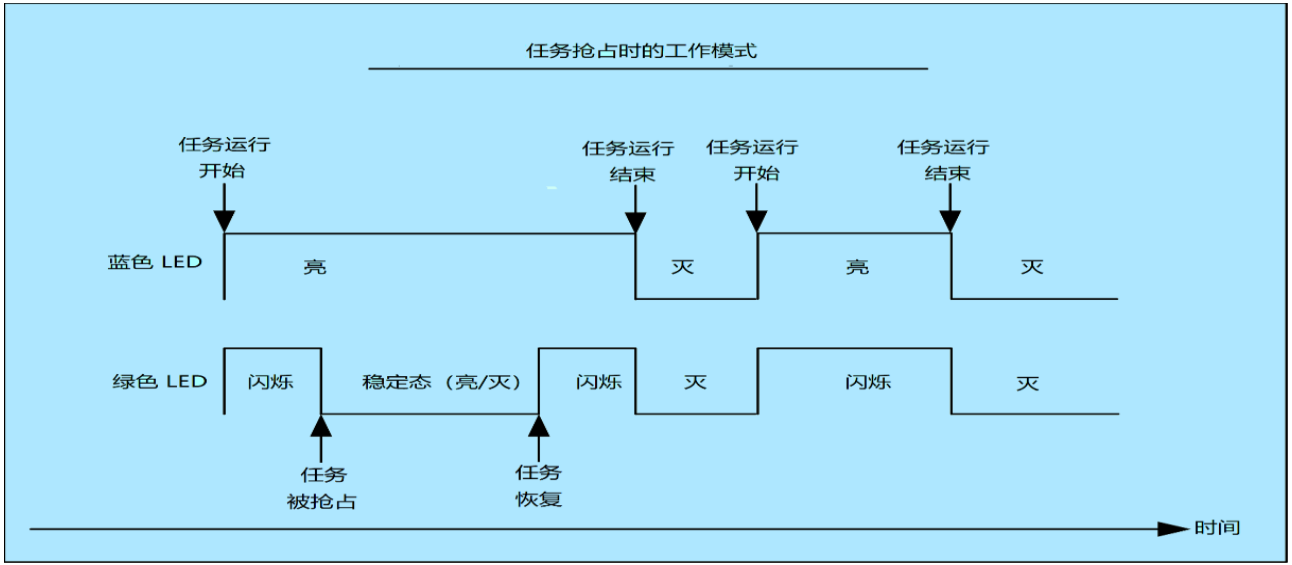
绿色LED任务正常工作模式（未发生任务抢占时）行为时序



绿色LED灯指示任务执行；蓝色LED灯指示任务未挂起的周期（即任务处于活动或就绪状态）

内核实验4-任务行为时序（2）

绿色LED任务被抢占时行为时序



在抢占期间绿色LED灯的状态将保持不变（点亮或熄灭）

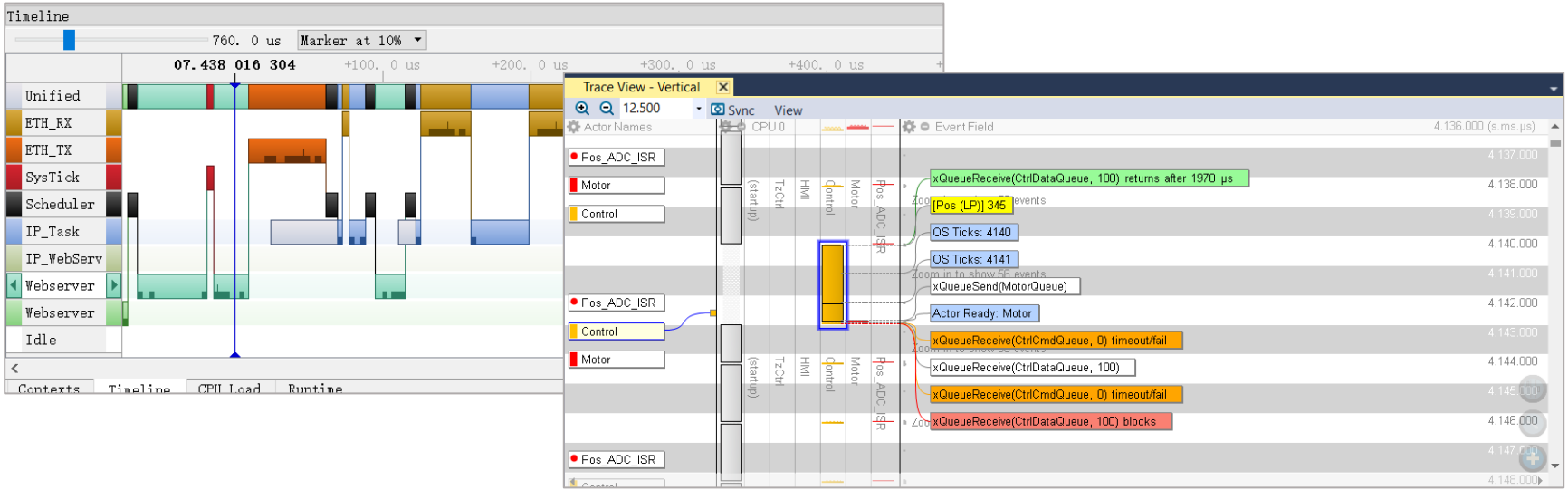
该模式仅存在于任务被抢占期间，通过观察此现象，可以获知抢占发生。

内核实验4-总结

- 保证任务满足其时间要求的唯一渠道是设置任务优先级为最高值。
- 低优先级任务如果被抢占，其实际运行时间（从开始到完成）将长于指定时间。
- 每次执行时间可能都有所不同，具体运行时间取决于抢占何时发生。
- 如果高优先级任务占用较长的执行时间，它会破坏整个系统的时间特性。
- 设计关键嵌入式软件，需要：
  - 详细定义时间要求
  - 仔细规划调度，并在可能的情况下预测性能
  - 使用任务分析工具精确分析运行时代码发生的情况
- 高响应性的系统必须具有低处理器利用率

什么是RTOS可视化分析工具?

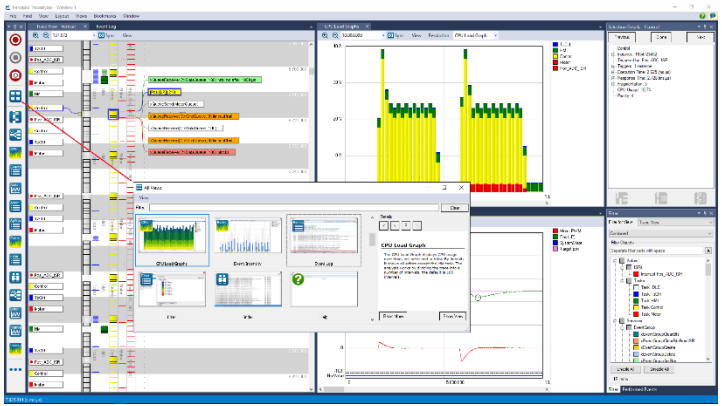
- 分析多任务嵌入式应用的软件工具
- 由PC端应用程序和目标端代码库组成
- 提供系统级的运行时可视化视图



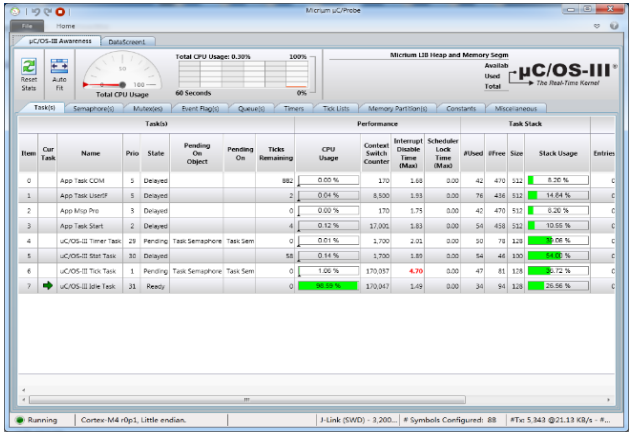
为什么要使用RTOS可视化分析工具?

- 验证RTOS系统的设计
  - 源代码并不能完全反映出多任务系统运行时的实时行为
  - 多任务系统的实时行为还取决与任务、中断、输入和他们的相互作用
- 减少故障排除的时间
  - 捕获不易复现的偶发性错误
  - 可以让很多嵌入式软件问题在短时间内解决
- 更好的优化实时系统
  - 精确的时间戳信息，视图与时序关联
  - 找出影响系统性能的瓶颈
- 更好的软件质量
  - 发现和避免潜在问题，如设计不当可能导致的CPU使用，调度和其他任务交互的相关问题

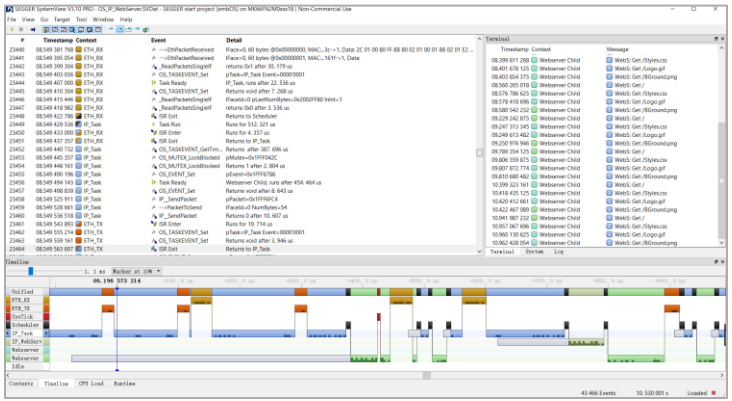
常见的可视化分析工具



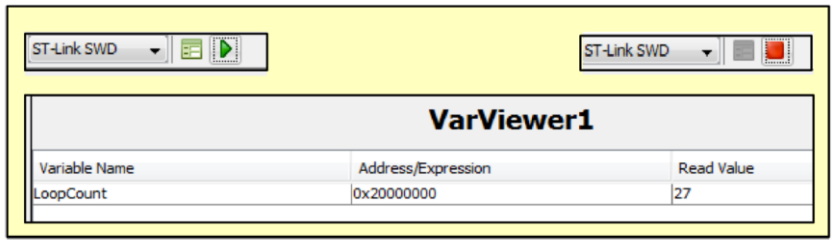
Tracealyzer



µC/Probe



SystemView



STM Studio

Tracealyzer可视化分析工具

- Tracealyzer是Percepio 公司开发的一款用于RTOS或基于linux的嵌入式软件系统的可视化跟踪工具
- 提供了30多种相互关联的运行时行为视图，包括任务调度、中断、任务之间的相互作用，以及从应用程序代码中记录的用户事件，且不需要额外的硬件。
- Tracealyzer作为传统调试的补充，提供更高层次的调试视图，适合理解典型的实时问题。

Tracealyzer支持的OS

FreeRTOS
Keil RTX5
Linux
On Time RTOS-32
ThreadX （Aruze RTOS)
µC/OS-III
OpenVX/Synopsys
VxWorks
Zephyr

实时操作系统（RTOS）已经存在了几十年了，但只是在近十年RTOS才在微控制器（MCU）中变得常见起来。RTOS提供了新的便于复杂嵌入式系统开发的一个抽象层。在RTOS上进行开发时，需要额外的工具和方法验证你的软件行为，还需要遵循RTOS应用设计的“最佳方法”验证你的软件，否则你的软件可能会变得不可靠、低效和难以调试。

《嵌入式实时操作系统——基于STM32Cube、FreeRTOS和Tracealyzer的应用开发》（原书第2版）关注RTOS应用开发的“最佳方法”，指导你规避常见的误区，教会你理解和控制软件的运行时行为，从而帮助你将新的创意用高效和可靠的方式转化为优秀的产品。

—— Dr.Johan Kraft Percepio公司CEO和CTO

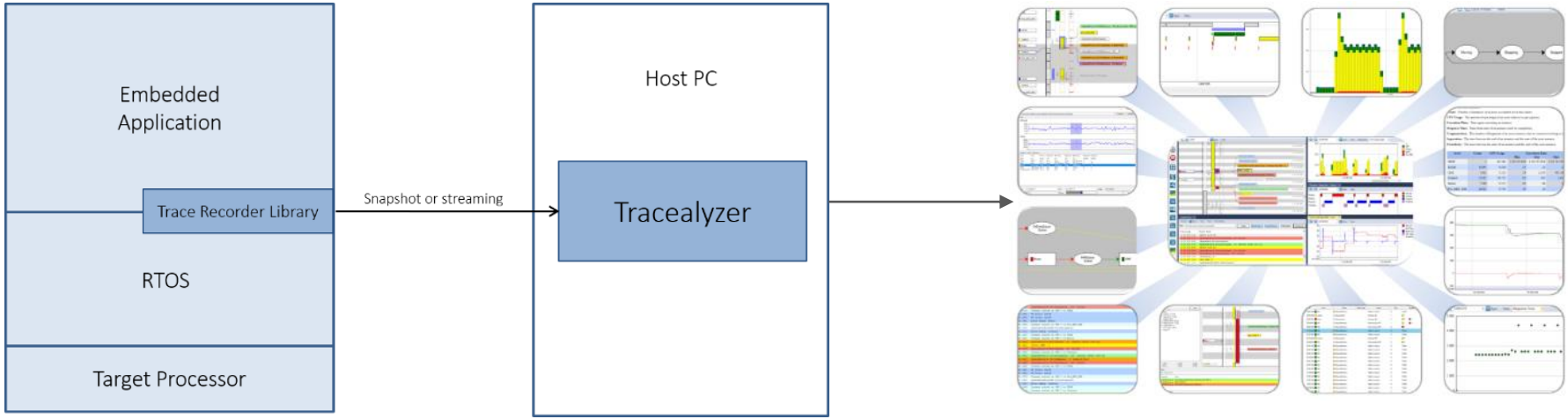
摘自：嵌入式实时多任务操作系统 清华大学出版社 封底

使用Tracealyzer可视化软件行为

- Tracealyzer实验1（第5章）
  - 演示Tracealyzer集成和配置
- Tracealyzer实验2-实验6（第6章）
  - Tracealyzer使用
  - 跟踪记录分析
  - 双任务运行时分析
  - 优先级抢占调度研究
  - FreeRTOS延迟函数分析
- Tracealyzer实验7（第7章）
  - 流模式跟踪记录分析
- Tracealyzer实验8-实验11（第8章）
  - 分析资源共享和任务间通信

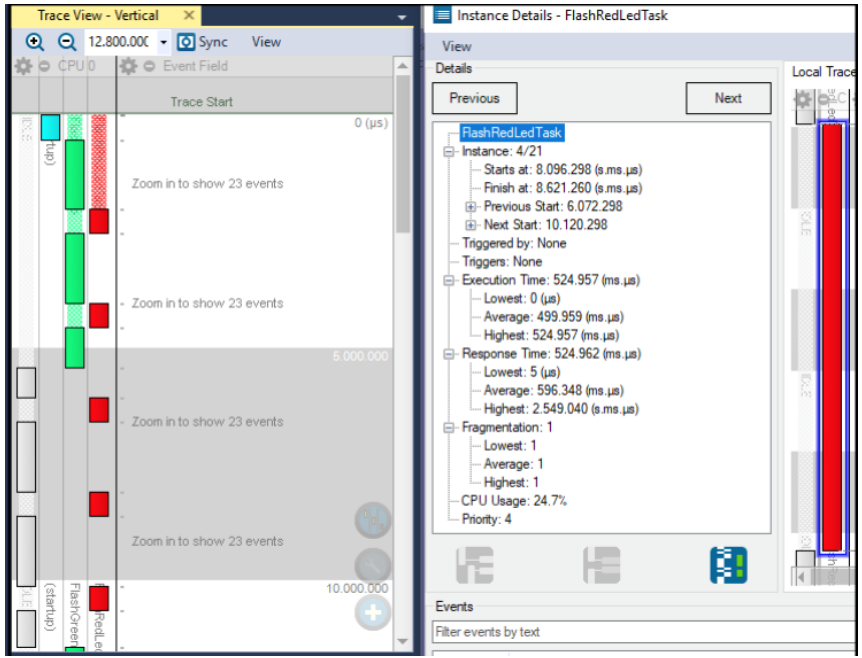
可视化抢占调度行为

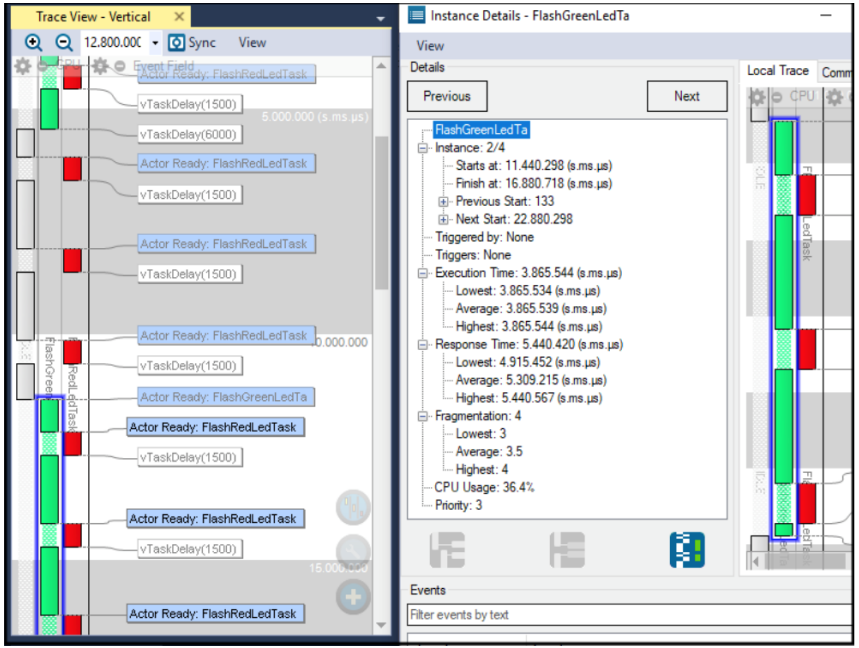
- Tracalyzer实验5
  - 实验代码基于内核实验4。
  - 使用Tracealyzer来观察采用优先级抢占调度策略的双任务设计的运行时行为
  - Tracealyzer快照工作模式
    - 从目标板的RAM读取跟踪缓冲区数据。



Tracealyzer实验5-跟踪记录分析（1）

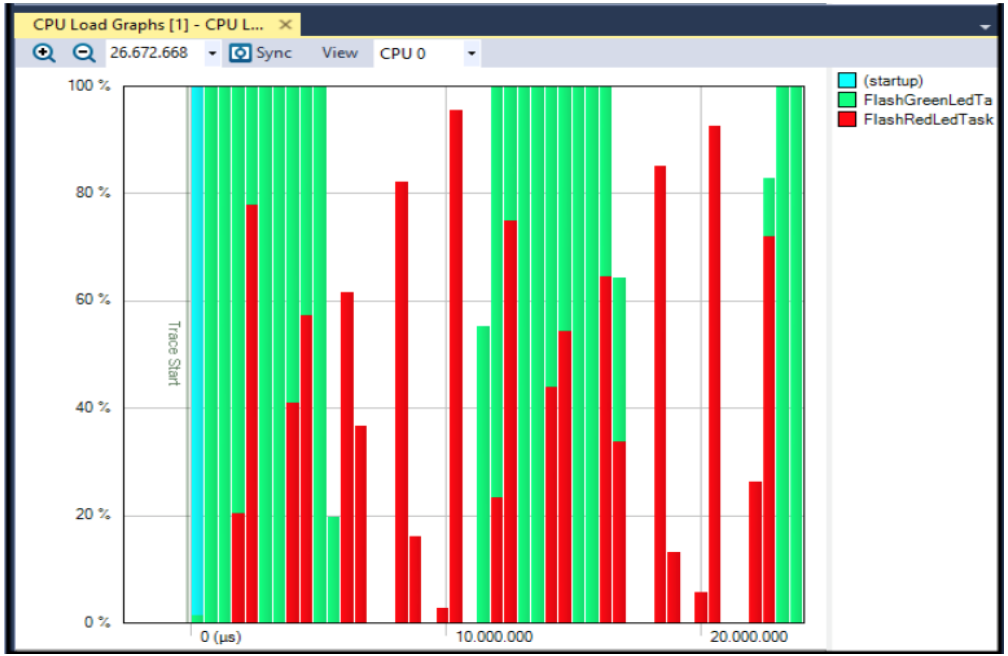
任务跟踪记录分析

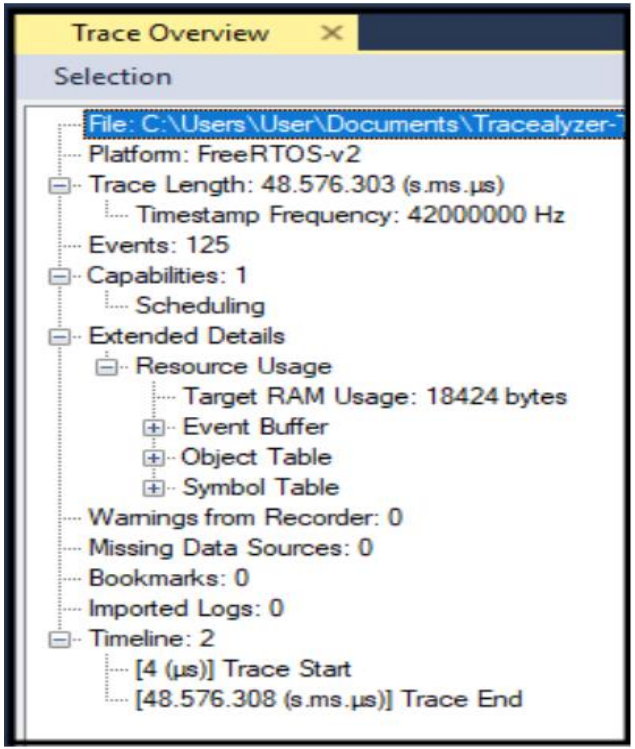




Tracealyzer实验5-跟踪记录分析（2）

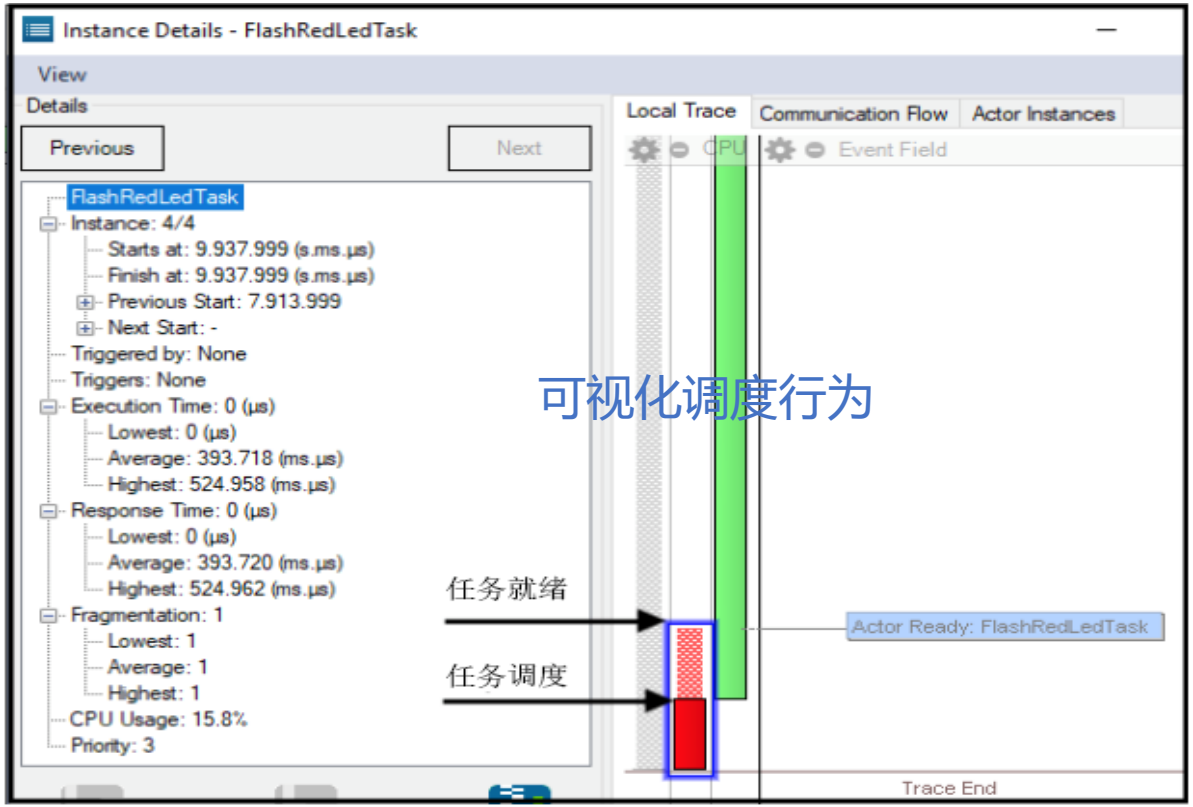
CPU负载及跟踪详细信息





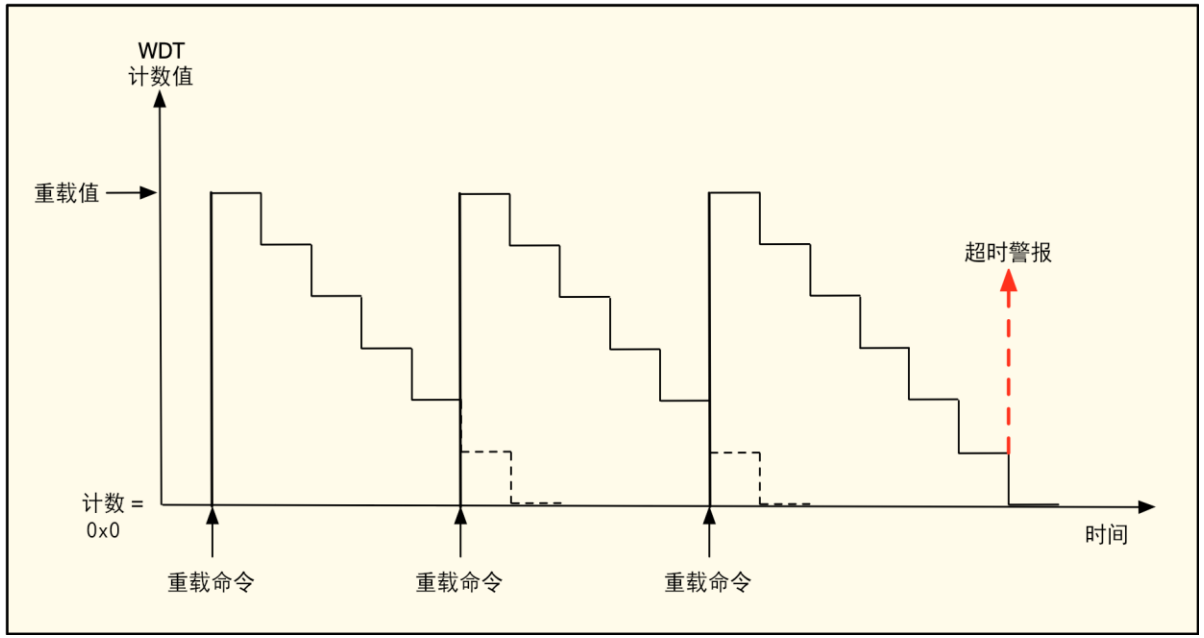
Tracealyzer实验5-跟踪记录分析（3）

就绪和调度的发生



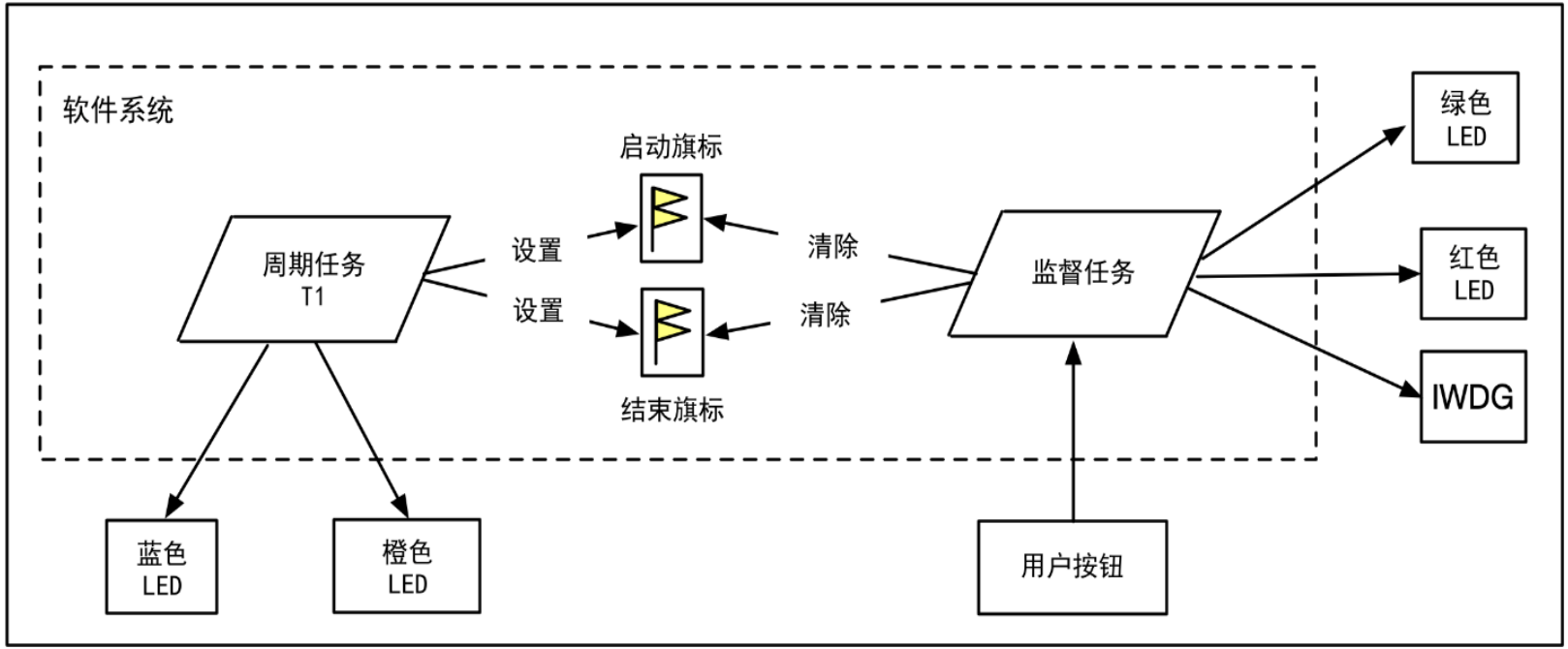
多任务系统中的故障检测

- 硬件机制
  - 通用定时器
  - 看门狗定时器
- 附加实验1-实验7（第10章）
  - STM32F4通用定时器使用
- 附加实验8-实验18
  - STM32F4看门狗定时器使用
- 附加实验19-实验22
  - 通用任务故障检测技术



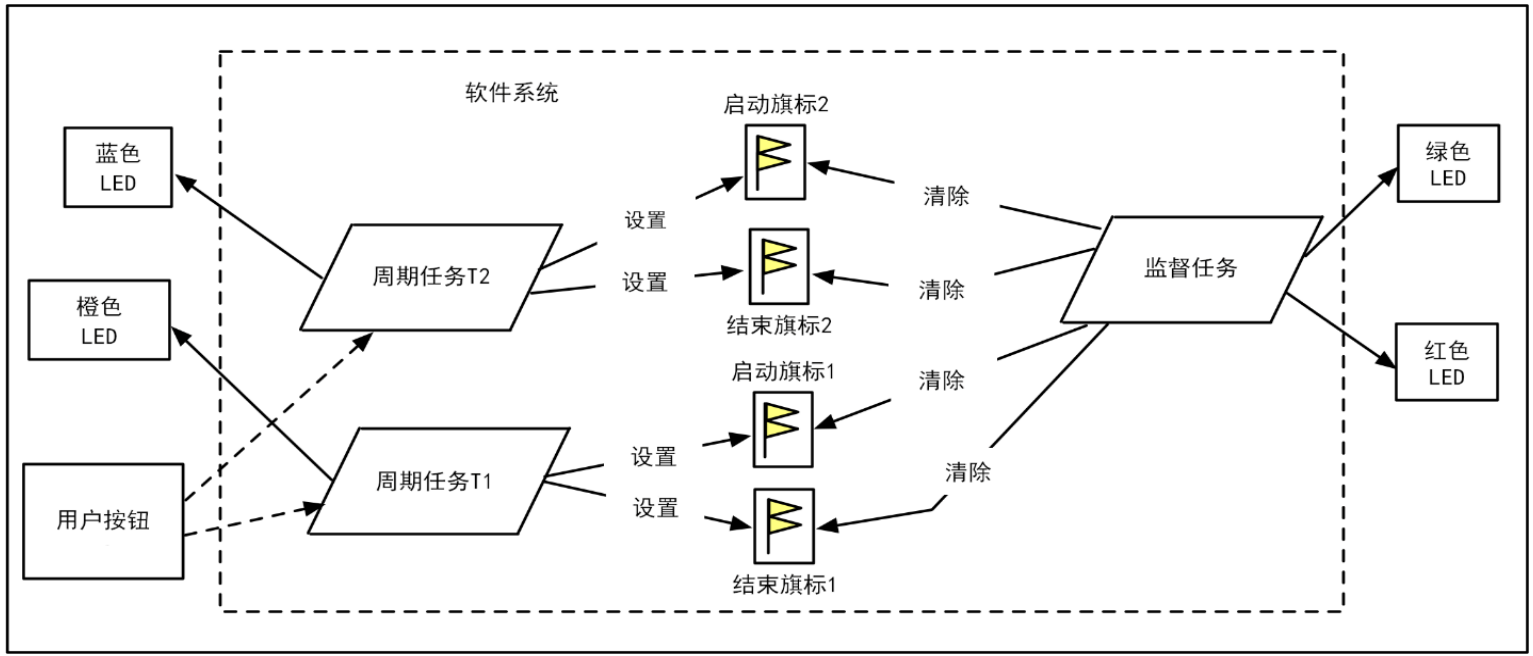
单周期任务的错误检测

- 实用的多任务设计必须同时检测应用任务和监督任务的故障
  - 用户任务错误检测
  - 监督任务错误检测



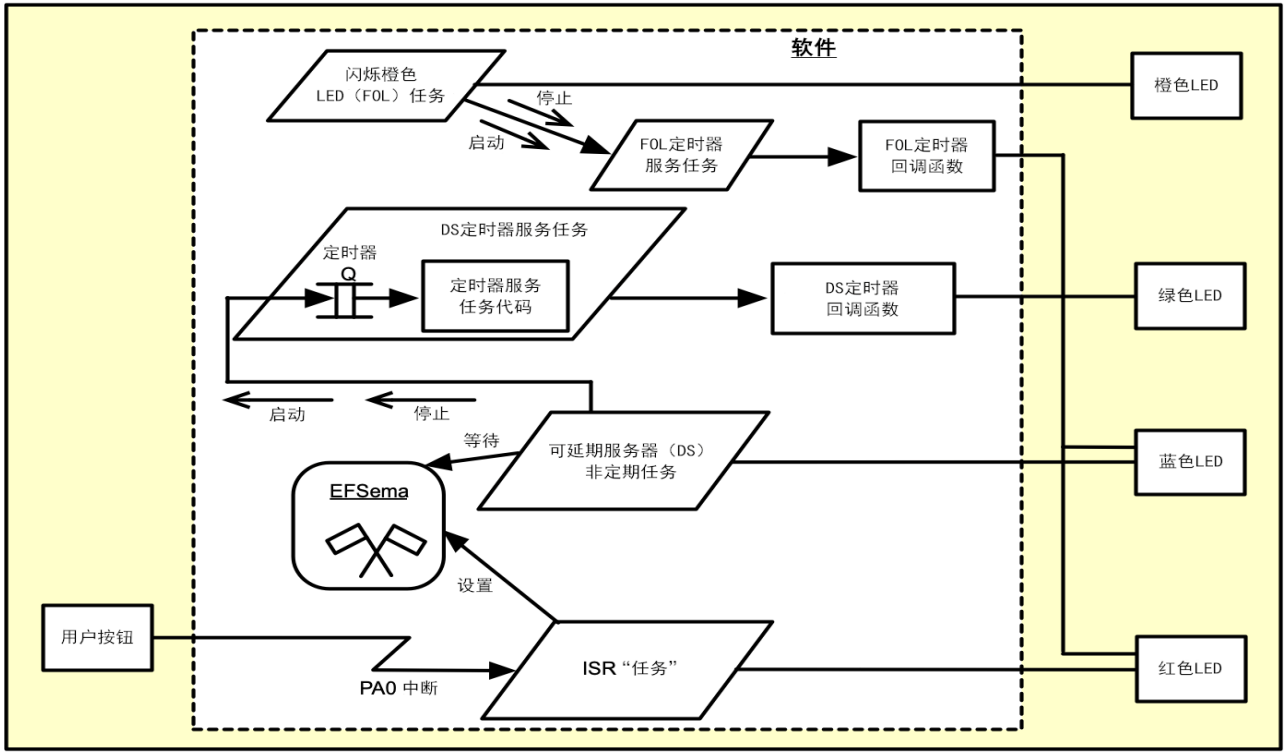
双周期任务的错误检测

应用任务故障检测



周期+非周期任务的错误检测

- 非周期任务错误
  - 非周期任务的不可预测性，执行错误检测前需确定任务的时间特性
    - 间隔时间
    - 执行时间
    - 阻塞时间



BMR
麦克泰技术

感谢您的聆听！

嵌入式实时操作系统学习群

图书资料和代码

http://hexiaoqing.net/publications

实时操作系统课程云课堂

为嵌入式与物联网行业提供产品、教育和服务