# PPT内容提取: pptx\_Trans\RT-Thread.pptx

## 幻灯片 1

第17章 RT-Thread线程管理

主讲人：李广宇

2025年4月

--------------------------------------------------

## 幻灯片 2

KEY POINTS OF CHAPTER

01

RT-Thread线程概念及管理方法

02

RT-Thread线程工作机制

03

本章概要

RT-Thread线程应用方法及实例

--------------------------------------------------

## 幻灯片 3

1

RT-Thread线程概念与管理方式

Section 17.1

--------------------------------------------------

## 幻灯片 4

17.1.1 线程概念

模块化编程是嵌入式系统设计的基本思想之一，通常将一个大的任务分解为多个简单易解决的小任务。线程是任务的实现载体，是RTT操作系统中最基本的调度单位。

--------------------------------------------------

## 幻灯片 5

17.1.2 线程管理方式及特点

线程管理的主要功能是对线程进行管理和调度，以实现不同线程的快速切换，达到多线程同时运行的目的，实际上同一时刻MCU只能运行一个线程。

--------------------------------------------------

## 幻灯片 6

线程调度由线程调度器完成，RT-Thread的线程调度器是抢占式：保证最高优先级的线程优先运行。

(1) 无线程运行时，从就绪线程列表中查找最高优先级线程运行。

(2) 低优先级线程使高优先级的线程满足运行条件，当前线程停止运行，高优先级的线程运行。

(3) 中断服务程序使高优先级的线程满足运行条件，中断完成时，被中断的线程挂起，高优先级线程运行。

线程切换时，调度器先将当前线程上下文信息保存，当再切回到这个线程时，调度器将该线程的上下文信息恢复。

17.1.2 线程管理方式及特点

--------------------------------------------------

## 幻灯片 7

2

RT-Thread线程工作机制

Section 17.2

--------------------------------------------------

## 幻灯片 8

17.2.1 线程控制块

线程控制块是操作系统用于管理线程的一个数据结构，存放了线程的优先级、线程名称、线程状态、链表结构、线程等待事件集合等信息，在RT-Thread中，线程控制块由结构体struct rt\_thread表示，指向线程控制块的指针称为线程句柄，用rt\_thread\_t表示。

--------------------------------------------------

## 幻灯片 9

17.2.1 线程控制块

线程控制块

线程句柄

结构体：

struct rt\_thread tid1;

tid1.name = “tid1”

结构体指针：

rt\_thread\_t tid2;

tid2->name = “tid2”;

--------------------------------------------------

## 幻灯片 10

17.2.2 线程重要属性

线程名称

线程名称即线程的名字，由用户命名，命名规则遵循C语言变量命名规则，通常以字母开头，线程名称的最大长度由 rtconfig.h 中的宏 RT\_NAME\_MAX 指定，多余部分会被自动截掉，默认长度为8位。

线程入口函数

线程入口函数是线程实现预期功能的函数，线程的入口函数由用户设计实现，有无限循环和顺序执行或有限次循环两种模式，在创建线程或初始化线程时可以传入参数。

--------------------------------------------------

## 幻灯片 11

17.2.2 线程重要属性

无限循环模式

void thread\_entry(void\* paramenter)

{

while (1)

{

/\* 等待事件的发生 \*/

/\* 对事件进行服务、进行处理 \*/

}

}

注意：调用延时函数或者主动挂起。

顺序执行或有限次循环模式

static void thread\_entry(void\* parameter)

{

/\* 处理事务 #1 \*/

…

/\* 处理事务 #2 \*/

…

/\* 处理事务 #3 \*/

}

注意：简单的顺序语句、do whlie() 或 for()循环

--------------------------------------------------

## 幻灯片 12

17.2.2 线程重要属性

线程栈

RT-Thread 线程具有独立的栈，当进行线程切换时，会将当前线程的上下文信息保存在栈中，当线程恢复运行时，再从栈中读取上下文信息，进行恢复。

线程栈大小可根据实际情况设定，对于资源相对较大的 MCU，可以适当设计较大的线程栈，对于资源较小的MUC可以在初始时设置较大的栈，如1K 或 2K 字节，然后在 FinSH 中用 list\_thread 命令查看线程运行过程中使用栈的大小，加上适当的余量形成最终的线程栈大小。

--------------------------------------------------

## 幻灯片 13

17.2.2 线程重要属性

线程优先级

线程的优先级表示线程被调度的优先程度，每个线程都具有优先级，应给重要的线程赋予较高的优先级，增大其被调度的可能。

对于 ARM Cortex-M 系列，普遍采用 32 个优先级。最低优先级默认分配给空闲线程（31），用户一般不使用。在系统中，当有比当前线程优先级更高的线程就绪时，当前线程将立刻被换出，高优先级线程抢占处理器运行。

--------------------------------------------------

## 幻灯片 14

17.2.2 线程重要属性

时间片

当线程优先级相同时，时间片才起作用，系统对优先级相同的就绪态线程采用时间片轮转算法进行调度，即线程轮转执行相应个系统节拍（1ms）。

--------------------------------------------------

## 幻灯片 15

17.2.2 线程重要属性

线程状态

对于单核MCU，同一时刻只允许运行一个线程，操作系统会自动根据线程运行的情况动态地调整线程状态。RT-Thread中线程共有五种状态。

获取资源得不到或者延时

强制结束（一般不用）

自然结束

获取到资源，或者延时时间到

启动

创建

--------------------------------------------------

## 幻灯片 16

17.2.2 线程重要属性

调用函数rt\_thread\_create/init创建/初始化的线程处于初始态；初始态线程调用函rt\_thread\_startup进入就绪态；就绪态线程被调度器调度后进入运行态；处于运行状态的线程调用rt\_thread\_delay，rt\_sem\_take，rt\_mutex\_take，rt\_mb\_recv等函数或者获取不到资源时，将进入挂起态；处于挂起态的线程等待超时依然未能获得资源或由于其它线程释放了资源，将返回就绪态。挂起态的线程调用rt\_thread\_delete/detach函数，将转换为关闭态；运行态的线程运行结束时，会在线程的最后部分执行函数rt\_thread\_exit，将状态改为关闭态。

--------------------------------------------------

## 幻灯片 17

17.2.2 线程重要属性

错误代码

--------------------------------------------------

## 幻灯片 18

17.2.3 系统线程

根据线程创建者，将线程分为系统线程和用户线程两类，系统线程是由RT-Thread内核创建的线程，用户线程是由应用程序调用线程管理接口创建的线程。

RT-Thread中的系统线程有空闲线程和主线程。

空闲线程

空闲线程（idle）是系统创建的最低优先级的线程，线程状态永远处于就绪态。当系统中无其它就绪线程时，调度器将调度空闲线程，它通常是一个死循环，且永远不能被挂起。

--------------------------------------------------

## 幻灯片 19

17.2.3 系统线程

空闲线程主要用于回收被删除线程的资源，如某线程运行完毕，系统将自动执行 rt\_thread\_exit函数，先将该线程从就绪队列中删除，再将该线程的状态更改为关闭态，然后挂入僵尸队列（资源未回收，处于关闭态的线程队列）中，最后由空闲线程回收该线程的资源。

此外，空闲线程提供了接口来运行用户设置的钩子函数，适合钩入功耗管理、看门狗喂狗等工作。

--------------------------------------------------

## 幻灯片 20

17.2.3 系统线程

主线程

系统启动时会创建main线程，它的入口函数为main\_thread\_entry，用户的应用入口函数 main就是从这里真正开始的，系统调度器启动后，main线程就开始运行，用户可以在main函数里添加自己的应用程序初始化代码。

--------------------------------------------------

## 幻灯片 21

3

RT-Thread线程应用方法及实例

Section 17.3

--------------------------------------------------

## 幻灯片 22

17.3.1 线程管理方式

--------------------------------------------------

## 幻灯片 23

17.3.2 动态线程应用步骤

最长8个字符

--------------------------------------------------

## 幻灯片 24

17.3.3 线程应用实例

利用线程实现状态指示灯，系统启动后，LED\_R先以0.5秒的间隔闪烁3次，然后进入正常运行状态，LED\_R以1秒的间隔闪烁。

打开RT-Thread Studio，基于开发板创建模板工程

--------------------------------------------------

## 幻灯片 25

17.3.3 线程应用实例

利用线程实现状态指示灯，系统启动后，LED\_R先以0.5秒的间隔闪烁10次，然后进入正常运行状态，LED\_R以1秒的间隔闪烁。

--------------------------------------------------

## 幻灯片 26

17.3.3 线程应用实例

--------------------------------------------------

## 幻灯片 27

谢谢观看

ADD YOUR SUBTITLE HERE

主讲人：李广宇

2025年4月

--------------------------------------------------