

RT-Thread 介绍

RT-Thread 历程回顾



RT-Thread的十年收获

软件生态

- 多种编译工具、工具链完善，可快速开发部署
- 各类应用框架，丰富的第三方软件支持

行业/客户

- 应用于能源、医疗、车载等众多高可靠性行业
- 获上百家各行业知名企业采用

硬件支持

- 商用支持所有主流MCU架构
- 几乎支持市面所有主流芯片方案

RT-Thread Nano版本

- 极简版本，业内最小。仅3kB Flash, 1.2kB SRAM占用

- 获得ARM官方认可，以MDK Pack方式内置于Keil MDK中
- 适合入门级32位 ARM MCU使用；理论覆盖任意Cortex-M0/0+, M3/4, M7芯片
- 无缝切换到RT-Thread标准版本



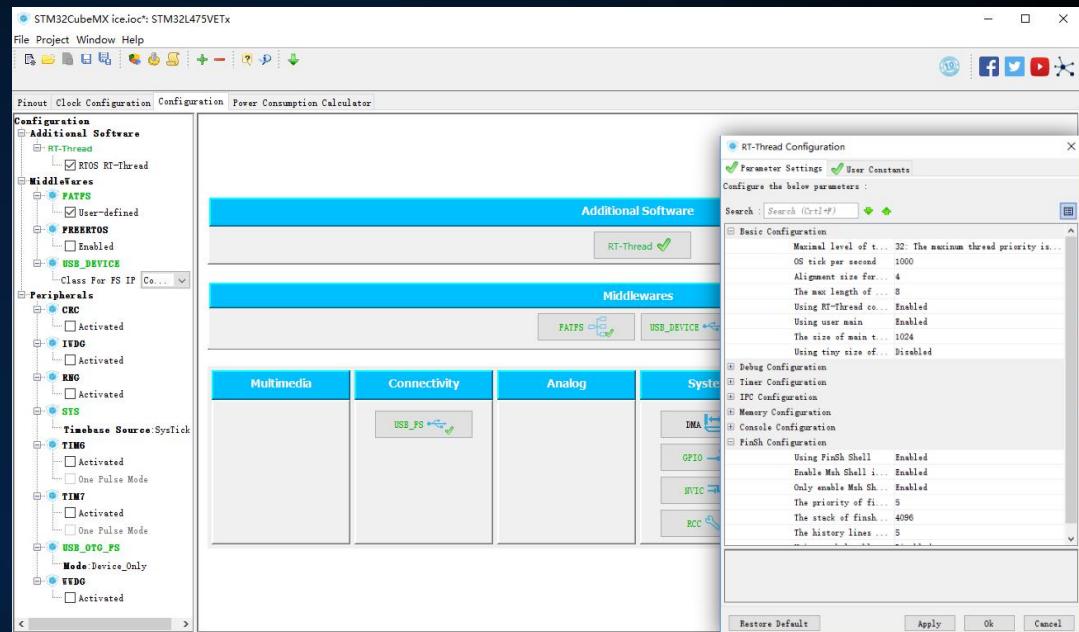
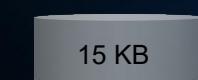
CubeMX中的RT-Thread Nano

使用STM32芯片更省心，直接基于ST CubeMX工具选择是否使用RT-Thread

- ◆ 生成Keil MDK、IAR工程；
- ◆ 与STM32 HAL紧密贴合；
- ◆ 配置RT-Thread细节情况：
 - ◆ shell组件
 - ◆ 设备框架；

ROM占用

RAM占用



物联网操作系统定义

AI, 智能化人机交互

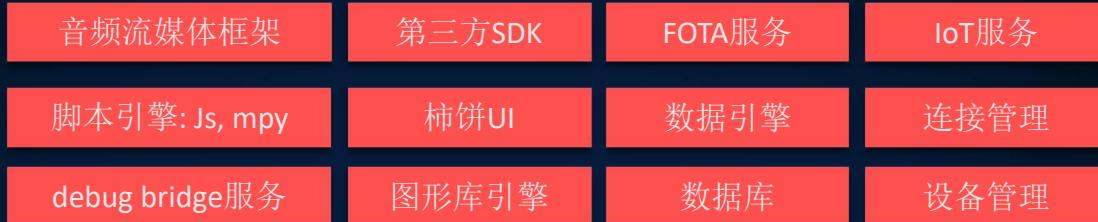


RT-Thread 架构图

应用层



软件包、中间件



RT-Thread 平台



RT-Thread 内核



安全框架

bootloader

基础组件

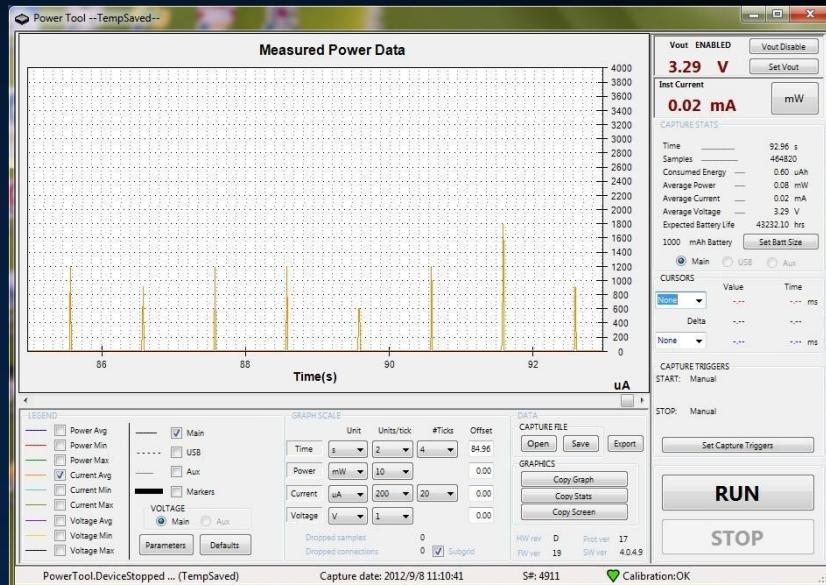
- ◆ **设备框架**: 名字化的对象化设备模型，上层应用与底层硬件设备无关
- ◆ **文件系统**: 虚拟文件系统，为上层应用提供统一的文件访问接口



特色组件: 低功耗组件

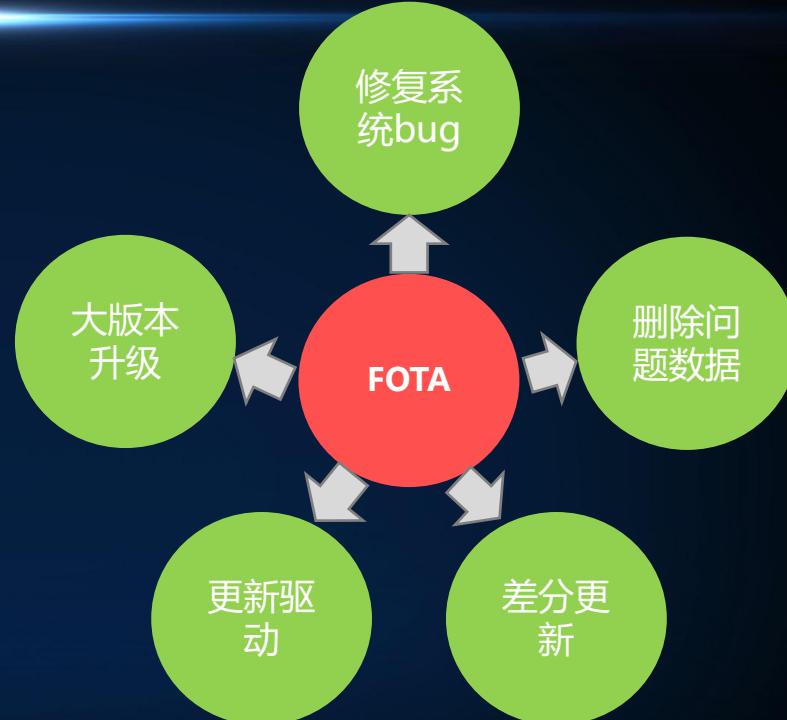
上层应用不需要关心底层功耗情况，系统自动进入休眠，达到最大省电的目的

- 系统空闲时休眠省电（支持睡眠模式，定时唤醒模式，停止模式）
- 系统激活工作时，根据程序设定值或基于芯片性能动态调节运行时频率



特色组件：通用Bootloader & FOTA

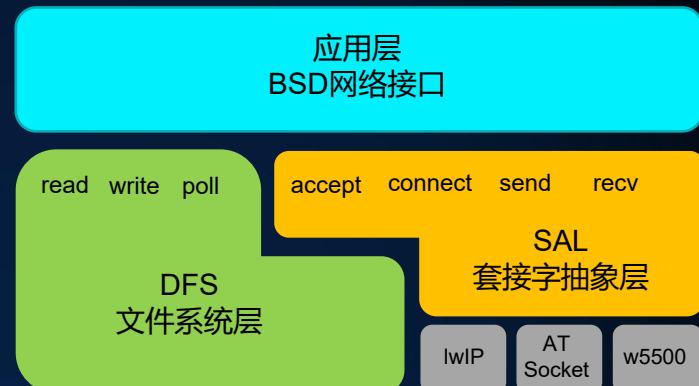
- 支持整包升级，不易出错、可靠性高；
- OTA固件与每一设备ID唯一绑定，防范固件被非法拷贝；
- 安全固件还原机制，保证系统不会变砖；
- 最新的安全加密算法、签名和校验、多重加密，保护每一次升级不会被恶意篡改；
- 支持断点续传、断电保护、智能还原、可回溯的安全机制，保证升级过程安全稳定。



特色组件: SAL套接字抽象层

SAL 真正实现了系统 (MCU+无线芯片/模块)
层面的跨平台软件开发 (ACS)

- 基于RT-Thread 的MCU控制器可以无缝接入各式各样的网络芯片或模块，上层应用无须改动
- RT-Thread支持的IoT软件包也可以“即装即用”
 - ◆ 抽象、统一多种网络协议栈 接口
 - ◆ 提供标准 BSD Socket API
 - ◆ 统一 fd (file descriptor) 管理方式

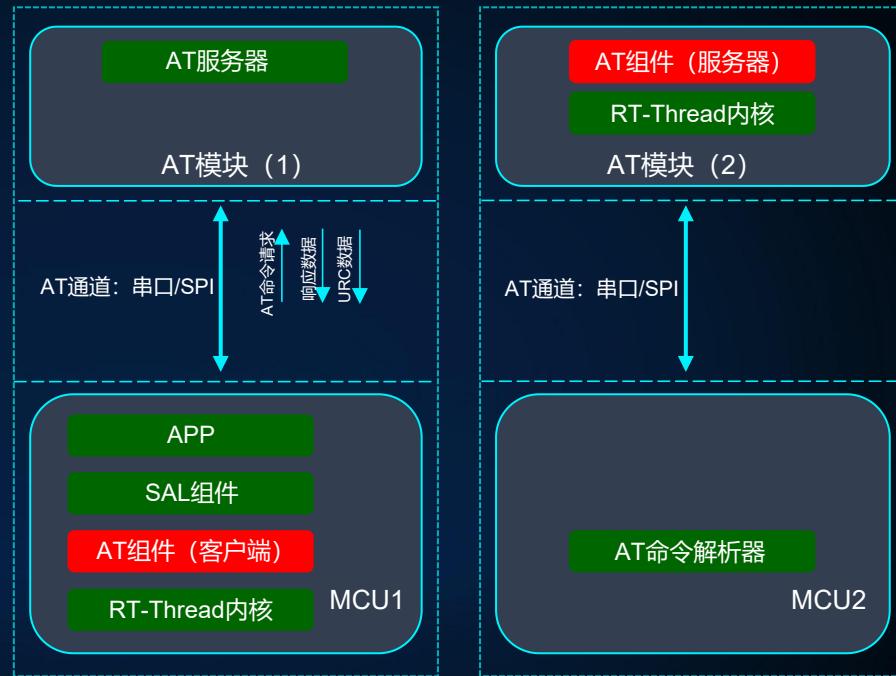


特色组件: AT组件

- ◆ 灵活的响应数据解析方式
- ◆ 完备的URC 数据的处理机制
- ◆ 简化 AT 命令的收发流程
- ◆ 提供标准的网络编程接口

已对接常用的 AT 模块

- 移远的M26 GPRS 模块
- 乐鑫的 ESP8266 WiFi 模块
- More to come...



特色组件: Persimmon UI

- 1 • 支持多点触摸操作，实现滑屏，拖拽，旋转，缩放等多种界面动画增强效果
- 2 • 包括按钮，图片框，列表，面板，card，wheel等基础控件，及窗口上悬浮带透明效果控件
- 3 • 使用类似signal/slot的方式，灵活的把界面事件映射到用户动作
- 4 • 支持TTF矢量字库，针对MCU优化的自定义图像格式，大幅提升图片加载和渲染速度



对称多核处理器支持

RT-Thread 4.0版本中加入对称多核处理器支持，完整的SMP调度，形成多核操作系统：

- ◆ 支持多个对称处理器核心：最大支持8个核心
 - ◆ CPU亲和度：BMP，可绑定任务到特定CPU集；
- ◆ 原有版本兼容性：对软件组件，应用都可复用，上层应用不需要关注多核运行情况；

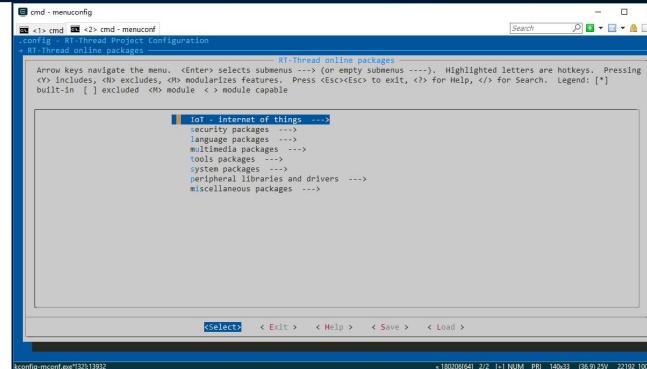


RT-Thread 软件包生态

软件包生态

经过一年来的发展，已经形成多达100+的各类软件包：

- 成为厂商，开发者模块化开发、设计的最佳选择；
- 为开发者的产品开发提供了强劲动力；



IoT

netutils, paho mqtt, CoAP,
web client, cJSON, ezXML,
WiFi/WICED, WiFi/Marvell



IoT/Cloud SDK

OneNet, Gagent, Ali-iotkit,
Azure IoT SDK



Security

mbedtls, TinyCrypt



Others

openmv, mupdf, libcsv,
optparse, quicklz, sample,
fastlz, miniLZO, zlib



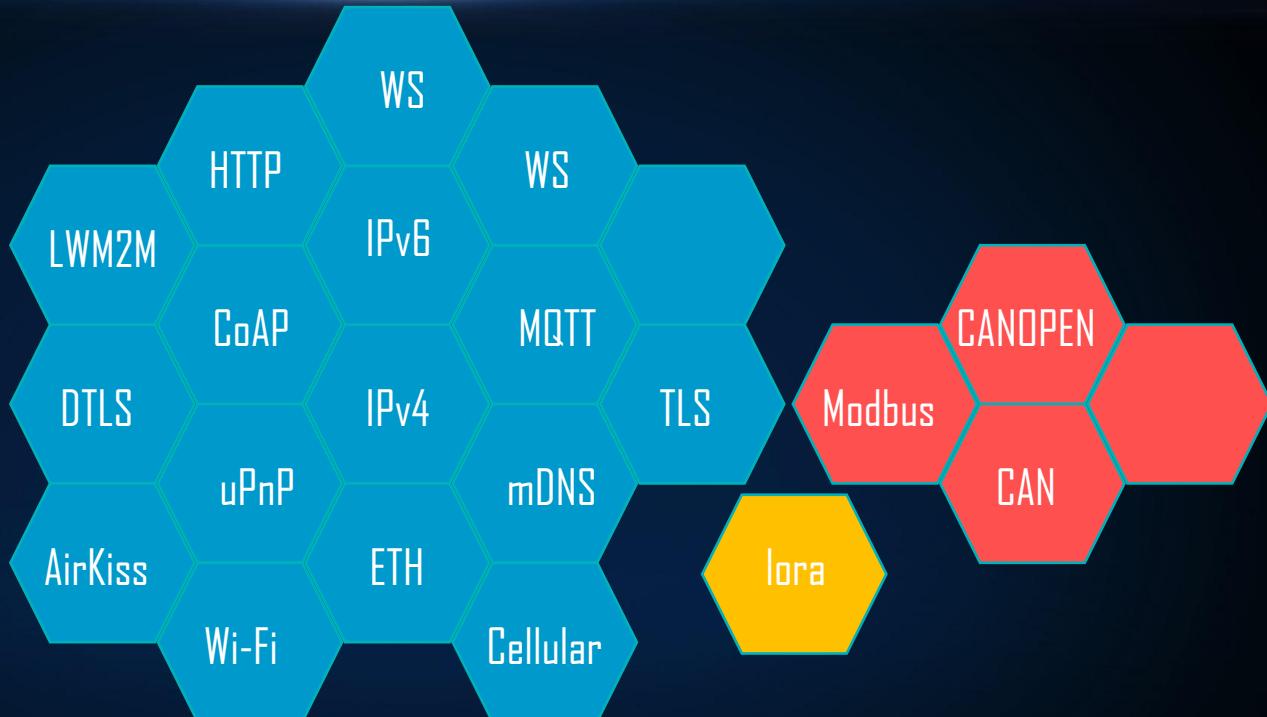
System

GUI Engine, Persimmon UI,
cairo, pixma, lwext4, fal,
SQLite, RTI, littlevgl, RT-Thread
YAFFS2 小而美的物联网操作系统

IoT连接软件包

包含丰富的网络支持

包括传输相关的安全、传
输底层支持。



RT-Thread云接入

云接入：更多对云的支持，开箱即可接入到云中！

可接入所有主流云平台



阿里云 IoT

提供 深度优化 的各云平台 SDK

开箱即用

资源占用低

连接可靠性高

定制化 SDK

云平台 SDK 模块化

功能可配置

高度可裁剪

支持 同时接入多个云

支持不同接入方式

AT 指令模组

WiFi SOC

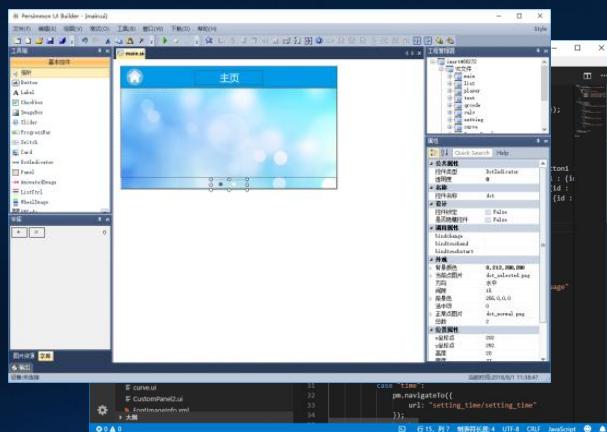
NB-IoT

2G/3G/4G

RT-Thread小程序

追求极致简单，最大化提升生成效率

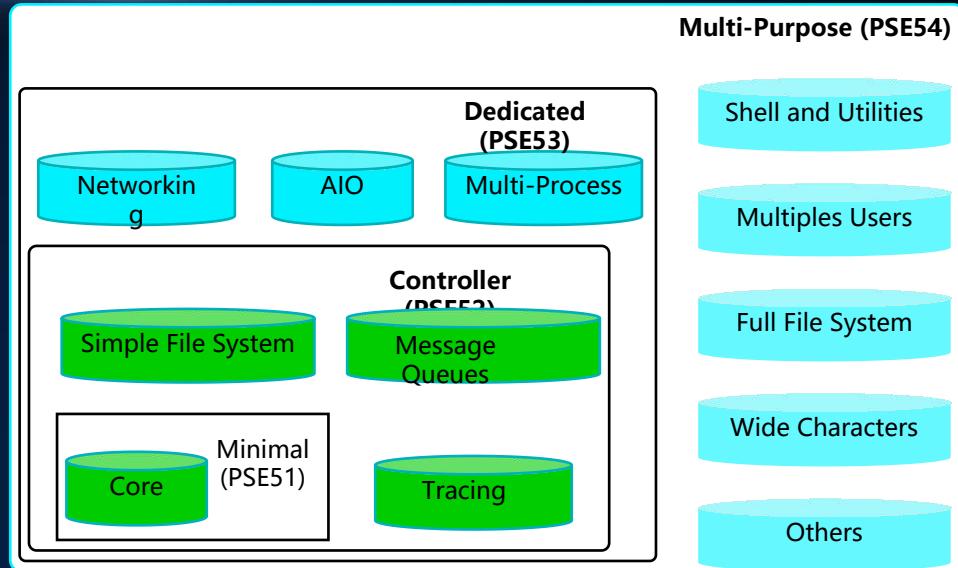
- ◆ 突破原有固件弊端：
 - ◆ 维护多套版本；
 - ◆ 程序复杂后，难以维护；
- ◆ 以JavaScript、MicroPython作为应用开发语言；
- ◆ 通过rdb打通PC、设备的交互障碍：
 - ◆ rdb over USB
 - ◆ rdb over 网络
 - ◆ 叠加vscode



POSIX及兼容性

- 在保持轻型POSIX层实现、配置可选的情况下，具备更好的兼容性，包括完整PSE52和网络相关的PSE53兼容能力。

- File I/O
- Net I/O，并支持网络、文件描述符联合 poll/select
- POSIX Threads
- POSIX signals
- terminos

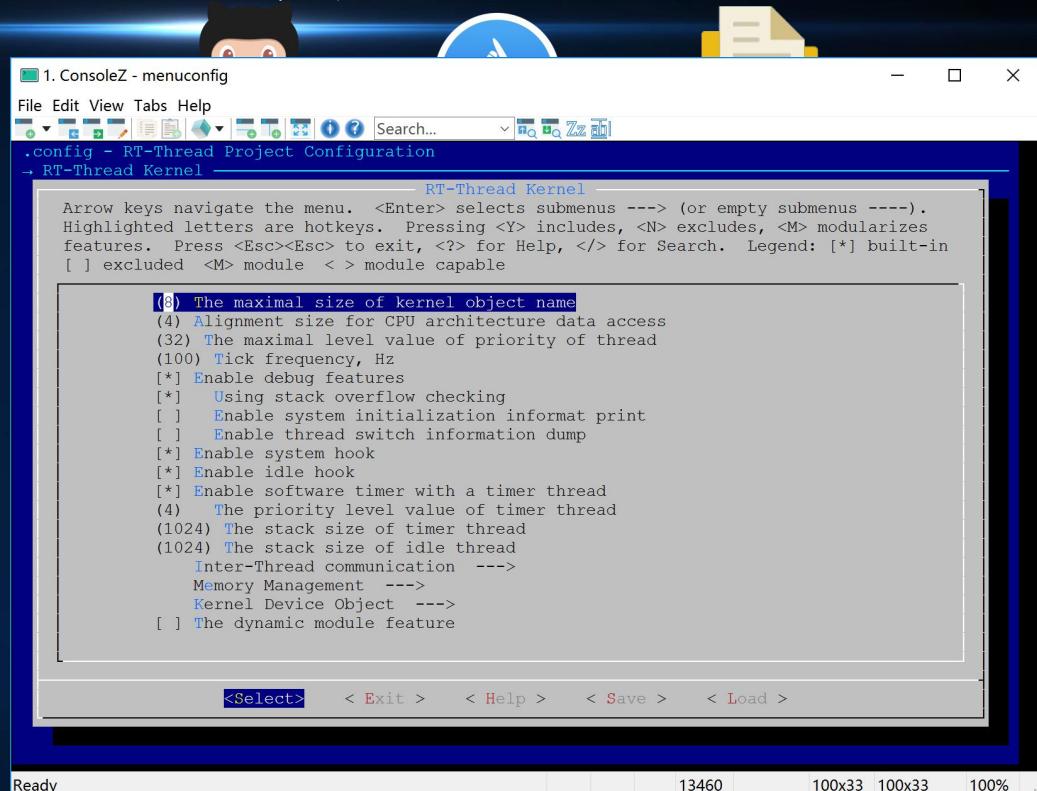


高度可伸缩



ENV工具

ENV工具



• menuconfig 配置

- 解决配置及配置依赖的问题
- 类似Linux Kconfig，具备Windows/Linux/MAC下相同的用户体验；

• packages 包管理器

- 在自己工程中灵活使用网络上的软件包 (github软件包、第三方软件包，或私有软件包)
- 扩展、丰富RT-Thread生态，让更多开发者享受分享的价值。

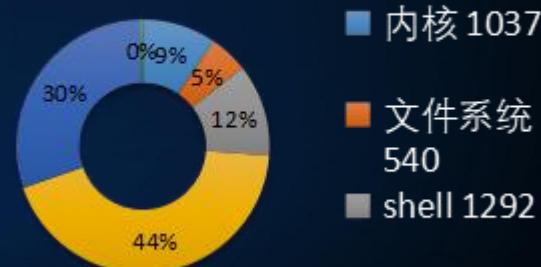
低资源占用

- 在网络应用场合，RT-Thread
资源占用情况
(ARM Cortex-M4)

CODE+RO Size: 71467 Bytes



RW+ZI Size: 10964 Bytes



HEAP

- total memory: 1048536
- used memory : 6224
- maximum allocated memory: 7448

易用和便捷开发

易用和便捷开发

架构清晰

C语言风格的内核面向对象的设计，完美的模块化设计

API简明齐全

代码注释清晰
便利应用二次开发

编译工具支持广泛

支持Keil, IAR, GCC开发环境

调试方便

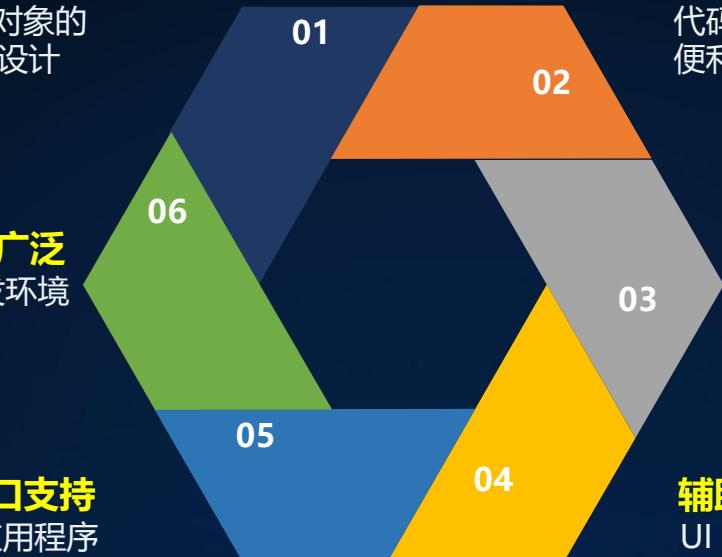
内置Shell调试工具，方便实时查看内核信息

POSIX接口支持

方便移植Linux应用程序

辅助工具

UI Builder, 配置器, 包管理器等
降低开发门槛，提升开发效率



学习和掌握RTOS

- 为什么学习RTOS?
- 如何学习RTOS?

裸机开发模式

后台:

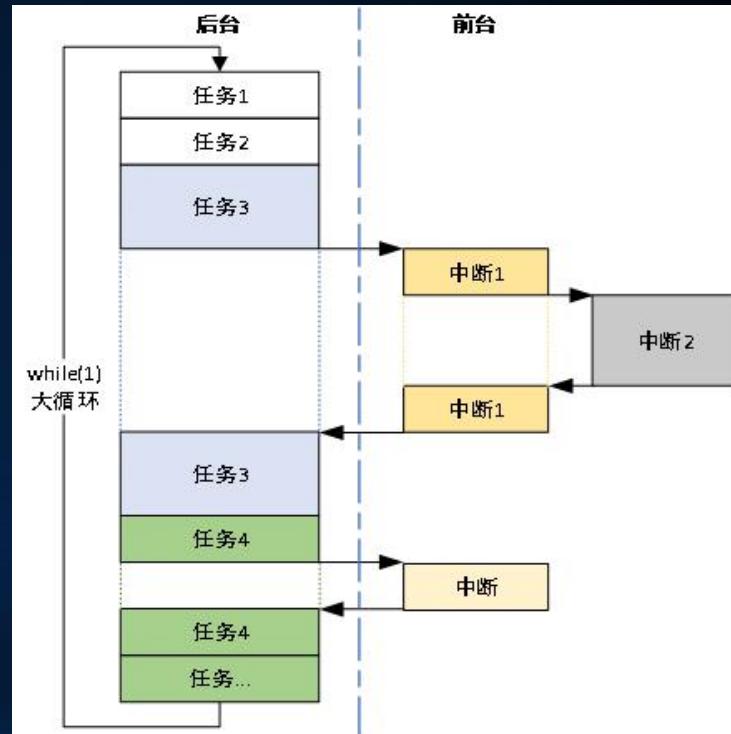
应用程序通常是一个无限的循环，在循环中，通过调用相应的处理函数，完成相应的操作，这部分可以看做为后台行为。

```
void main(void)
{
    init();
    while(1)
    {
        ADC_Read();
        SPI_Read();
        USB_Packet();
        LCD_Update();
        Audio_Decode();
        File_Write();
    }
}
```

前台:

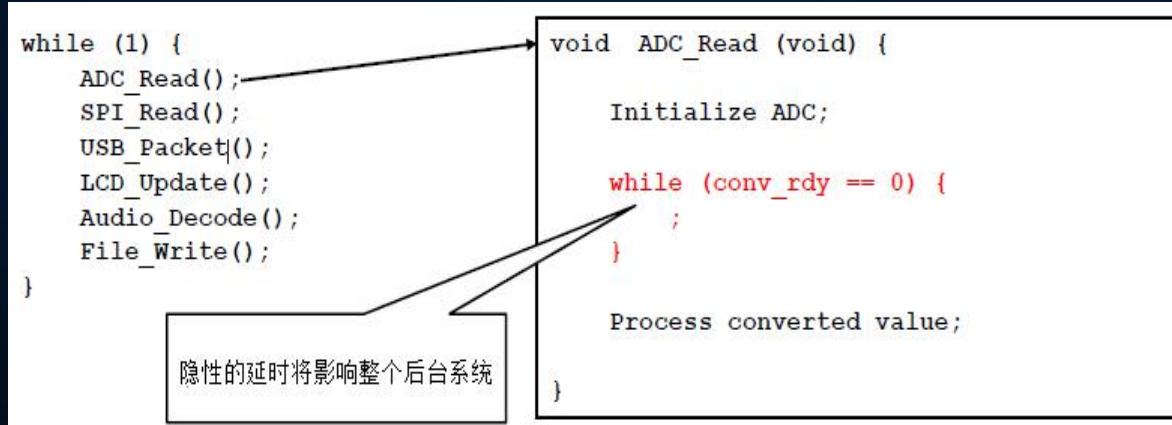
中断服务程序接收异步中断，来通知后台，后台收到中断请求后进行处理

```
void USB_ISR(void)
{
    Clear interrupt;
    Read packet;
}
```



裸机模式缺陷

- 并发性：程序并发工作效率低
- 实时性：功能复杂的情况下，实时性无法保证
- 可维护性：代码/维护困难，改变部分功能能影响整个系统代码，牵一发而动全身
- 可重用性：模块化程度低，软件可重用性差，总是重复造轮子



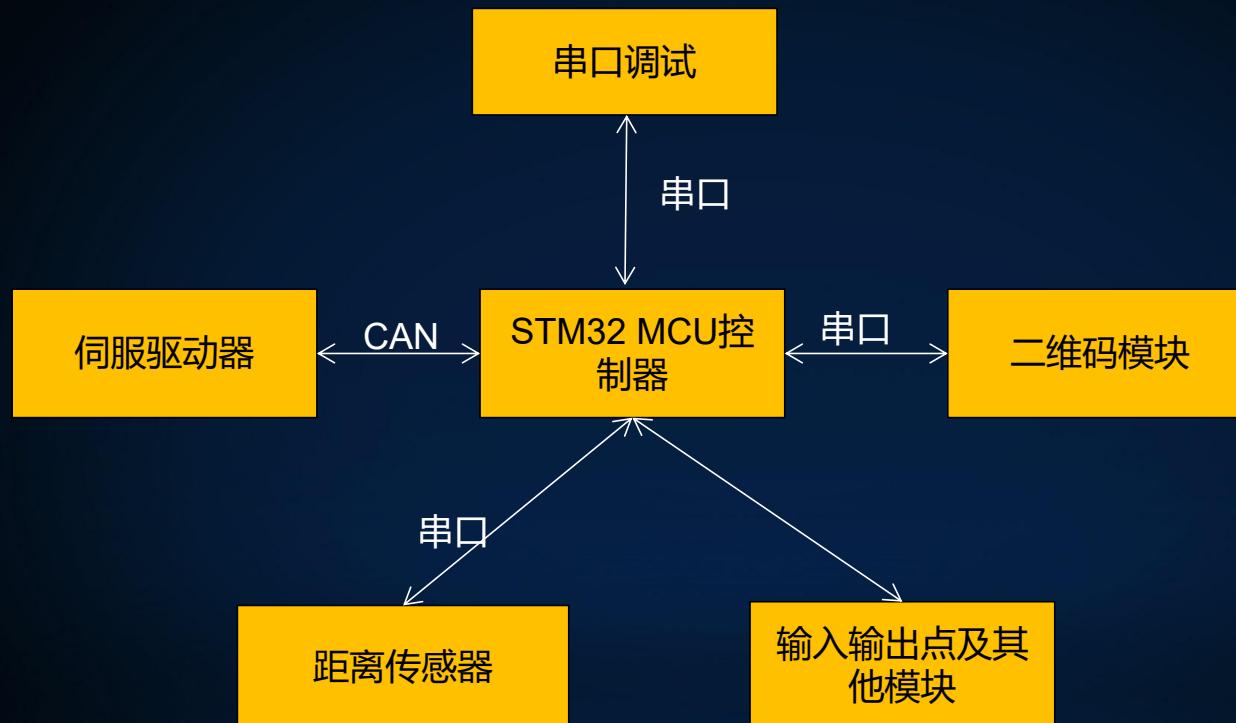
RTOS开发应用案例

潜入式AGV

- 全称 Automatic Guided Vehicle
- 用于电商仓库，将移动货架从仓库中的储存位搬运至拣选工位
- 用于工厂可以实现将物料从一个工位搬运至下一个工位



系统组成



系统组成

- AGV 中一共使用了3个 UART
 - 一个连接二维码模块，用于读取二维码信息
 - 一个距离传感器，用于读取障碍物距离信息
 - 最后一个用于串口调试
- AGV 中一共使用了4个伺服电机，控制器和伺服驱动器之间采用 CAN 总线通讯

多任务软件设计

运动控制线程

```
while(1){  
    等待接收事件；  
    if(接收障碍物信息){  
        避障处理  
    }  
    else if(接收定位信息)  
        控制运动  
}
```

避障线程

```
while(1){  
    检测障碍物距离信息；  
    if(检测到障碍物距离近){  
        通知运动控制线程处理  
    }  
}
```

二维码扫描线程

```
while(1){  
    读取二维码信息；  
    if(获得二维码信息){  
        计算定位信息  
        并通知运动控制线程  
    }  
}
```

需求扩展



软件设计

运动控制线程

```
while(1){  
    等待接收事件；  
    if(接收障碍物信息){  
        避障处理  
    }  
    else if(接收定位信息)    }  
        控制运动  
    else if(接收调度信息)  
        控制运动  
}
```

避障线程

```
while(1){  
    检测障碍物距离信息；  
    if(检测到障碍物距离近){  
        通知运动控制线程处理  
    }  
}
```

二维码扫描线程

```
while(1){  
    读取二维码信息；  
    if(获得二维码信息){  
        计算定位信息  
        并通知运动控制线程  
    }  
}
```

Wi-Fi调度线程

```
while(1){  
    读取WiFi通讯数据并解析；  
    if(运动控制命令){  
        通知运动控制线程  
    }  
}
```

组件开箱即用

CanFestival 组件

移植并配置好CanFestival之后，就可以通过读写MCU内部变量来控制伺服的位置、速度、加速度，读取当前位置、速度等信息

Protocol buffers 组件

Protocol buffers 是一种语言中立，平台无关，可扩展的序列化数据的格式，可用于通信协议，数据存储等。

RTOS给软件设计带来的好处



- 并发性：支持并发处理（实时性）
- 扩展性：容易加入新的功能，方便系统更新和维护
- 模块化：破解应用的复杂性
- 开发效率：组件拿来即用，不用重复造轮子

RTOS给公司和个人带来价值

对公司价值

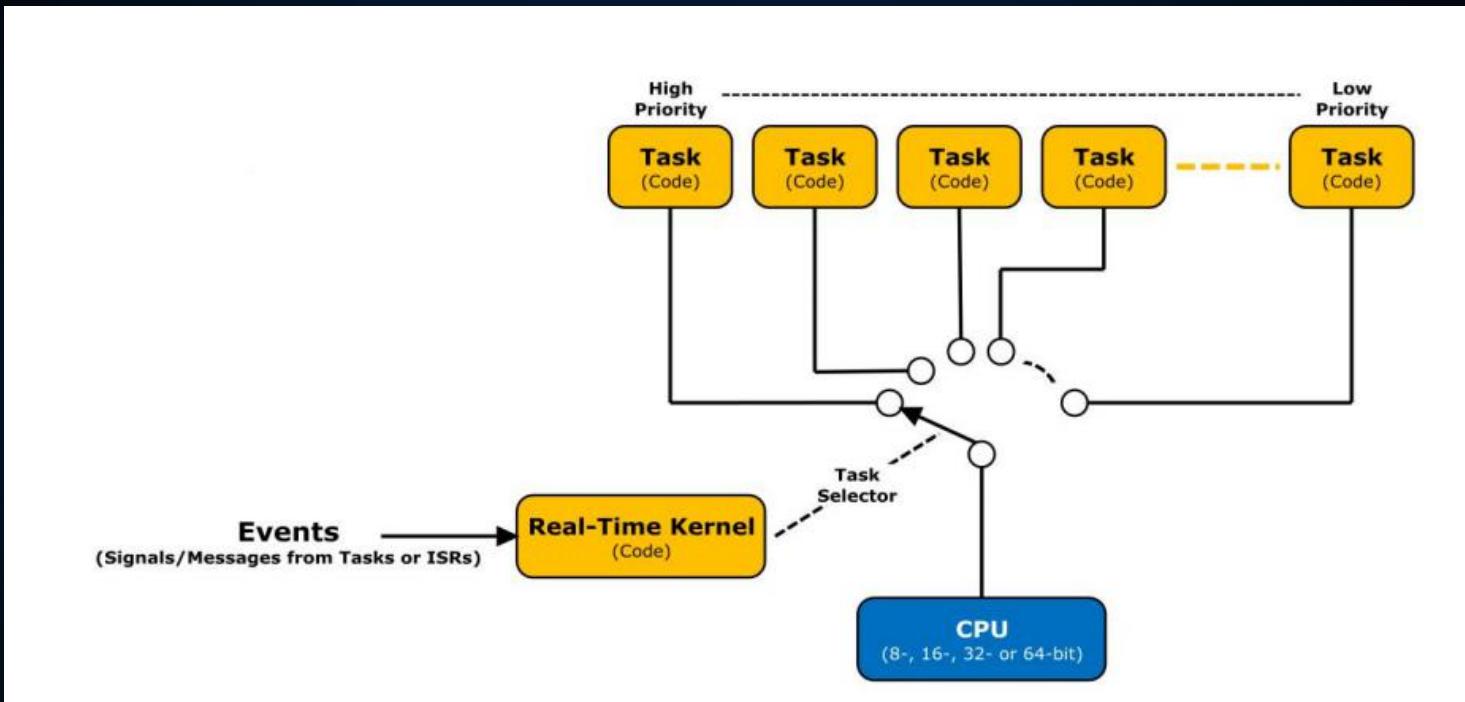
- 管理效率：模块化的开发，适合多人协作，提高了软件研发管理的效率。
- 研发质量、成本：众多的可重用组件（包括开源），降低了研发周期的同时，也提高了产品质量，为公司节省了研发成本
- 积累资产：不断积累的可重用软件组件，也是公司的无形资产。

对个人价值

- 增加技能，提升个人价值
- 能接触到更多高级软件组件，提高自己的知识广度与深度
- 不想把时间都浪费在重复造轮子上

- 为什么学习RTOS?
- 如何学习RTOS?

RTOS精髓：多任务系统



掌握RTOS内核核心要素

•事件驱动

- 中断机制和多任务
- 优先级抢占和时间片轮转调度

•资源共享

- 任务间通信和同步互斥
- 提供的机制有信号量、邮箱、消息队列、事件标志、互斥

学习RTOS方法

- **任务管理是重点**

- 掌握任务建立，调度和通信机制
- 掌握内存管理方式
- 学习RTOS 内核和硬件相关部分 - 中断和时钟管理

- **简单的驱动编写**

- 比如串口、GPIO 等基本外设
- 移植已经不是重点

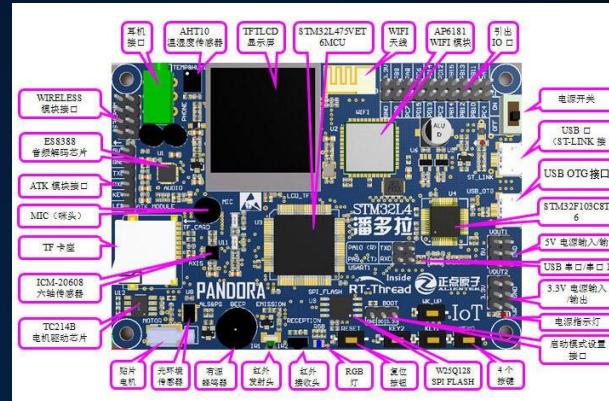
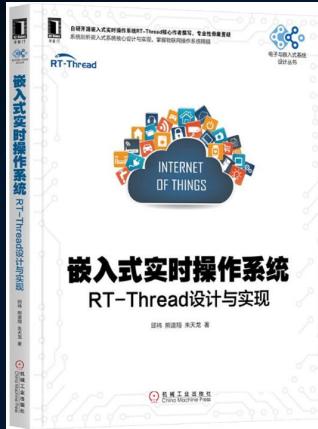
- **RTOS编程接口**

- RTOS接口标准
- 设备、文件系统、网络接口
- POSIX标准

- **RTOS 组件**

- 用到再学习

学习资料



资源中心

<https://www.rt-thread.org/document/site/>



谢谢！