RISC－V MCU 的 FreeRTOS 移植与应用开发

付元斌，张爱华，何小庆

（北京麦克泰软件技术有限公司，北京 100085）

摘要：具有相同的 RISC－V 指令集的处理器实现并不相同。本文将针对基于 RISC－V 开源指令集的处理器芯片 GD32VF103 MCU，介绍 FreeRTOS 在 IAR EWRISC－V 编译和开发环境下的移植过程。采用 RTOS 后，嵌入式系统很难监控系统的运行时行为、发现应用存在的问题，本文基于 Tracealyzer 分析工具直观地跟踪系统行为，分析系统中可能的错误，提高代码的鲁棒性。

关键词：Bumblebee 内核；Tracealyzer；FreeRTOS 移植；RISC－V 处理器

中图分类号：TP316.2　　**文献标识码**：A

Porting and Application Development of FreeRTOS Based on RISC－V MCU

Fu Yuanbin，Zhang Aihua，He Xiaoqing

（Beijing Microtec Research Software Technology Co. Ltd.，Beijing 100085，China）

Abstract：Processor implementations with the same RISC－V instruction set are not the same. This paper will introduce the migration process of FreeRTOS under IAR EWRISC－V environment for the processor chip GD32VF103 based on RISC－V open source instruction set. With the RTOS，it is difficult to monitor the runtime behavior of the system and find problems in the application. This article will use the Tracealyzer analysis tool to visually system behavior，analyze possible errors in the system，and improve the robustness of the code.

Keywords：Bumblebee Kernel；Tracealyzer；FreeRTOS porting；RISC－V MCU

0　引　言

RISC－V 是基于精简指令集（RISC）原则的开源指令集架构（ISA），具有良好的应用前景。目前 RISC－V 架构的处理器陆续发布，其生态环境及应用也在不断丰富，但相对于 ARM 架构，RISC－V 的应用还处于起步阶段，复杂的应用需要 RTOS 支撑。目前许多 RTOS，如 μC/OS－III、FreeRTOS、RT－Thread、Zephyr OS、embOS 等，都提供了对 RISC－V 处理器的支持。采用了 RTOS 后，传统的调试手段很难监控系统的实时行为，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，提高代码的可靠性。

本文使用兆易创新的 GD32VF103V－EVAL 开发板，基于 IAR EWRISC－V 编译和开发环境，介绍如何将 FreeRTOS 移植到 GD32VF103 上，并使用 Percepio 的 Tracealyzer 工具分析基于 FreeRTOS 的应用程序运行时行为。

1　GD32VF103 RISC－V MCU

GD32VF103 RISC－V MCU 基于 Nuclei Bumblebee 内核，芯片提供了 108 MHz 的主频，以及 16～128 KB 的片上闪存和 6～32 KB 的 SRAM，具有多个通用定时器和多通道 DMA 控制器。GD32VF103 MCU 的中断控制器（ECLIC）提供了多达 68 个外部中断并可嵌套 16 个可编程优先级，GD32VF103 MCU 支持多种外设如 UART、SPI、I^2C、GPIO 等。GD32VF103V－EVAL 评估板（见图 1）使用 GD32VF103VBT6 作为主控制器，板载 JTAG 接口以及丰富的外设资源。

RTOS 在 GD32VF103 上的移植仅使用了内核相关的资源。Nuclei Bumblebee 内核支持 RV32IMAC 指令集架构；特权模式支持机器模式和用户模式，可以实现 RTOS 与应用的隔离，提升软件安全；寄存器组包含 32 个通用寄存器、RISC－V 标准的状态寄存器以及内核自定义

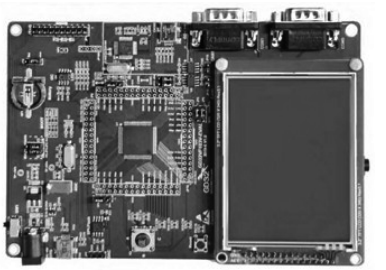

图 1　GD32VF103V－EVAL 评估板

的 CSR 寄存器。内核提供了计时器单元(Timer Unit,TIMER),产生的计时器中断和软件中断可以用于 RTOS 时钟节拍处理和任务切换。

Bumblebee 内核支持在 RISC-V 标准 CLIC 基础上优化而来的改进型内核中断控制器(Enhanced Core Local Interrupt Controller,ECLIC),用于管理所有的中断源。ECLC 的每个中断源可以配置为向量或非向量处理模式。向量模式中断被处理器内核响应后,处理器直接跳入该中断的向量入口存储的目标地址;非向量模式中断被处理器响应后,处理器直接跳入所有中断共享的入口地址。但中断响应时,处理器硬件不会保存上下文,需要软件实现。

2 FreeRTOS 和 Tracealyzer 介绍

FreeRTOS 是英国人 Richard Barry 2003 年发布的开源实时内核。FreeRTOS 支持超过 35 种 CPU 架构,是世界最受开发者欢迎的 RTOS,在 2017 年每 3 分钟就有一次下载。FreeRTOS 有一个系列软件,包括 FreeRTOS(开源版本)、OpenRTOS(授权版本)、SAFERTOS(安全版本)和 Amazon FreeRTOS(开源物联网操作系统),开源的 FreeRTOS 遵循 MIT license 模式。

在多线程软件系统中,RTOS 带来的复杂度让源代码和运行时行为的关联变得不那么明显,传统的调试器不足以理解系统的行为,特别是运行时行为。

Percepio 的 Tracealyzer 工具运行在 Windows 或 Linux PC 上,可用于目标系统运行 Linux、FreeRTOS、OpenRTOS、SAFERTOS、Wind River VxWorks、μC/OS-III 和 embOS 的 RTOS 应用行为分析。

2.1 FreeRTOS 的功能

FreeRTOS 支持抢占和时间片轮询两种任务调度方式,支持无限数量的应用任务;提供队列、信号量、互斥信号量、事件标志等内核机制,满足任务间同步及通信需求;此外,针对低功耗应用提供了 tickless 模式。最新版本 FreeRTOS 提供了针对 IAR 及 GCC 工具链的标准 RISC-V 处理器内核移植示例,支持 32 位及 64 位架构内核(RV32I 和 RV64I)。它包含了预配置的 OpenISA VEGAboard、SiFive HiFive 开发板、QEMU 模拟器以及用于 Microchip M2GL025 开发板的 Antmicro Renode 模拟器示例,可以扩展支持任何 RISC-V 处理器,但因为每种 RISC-V 处理器在微架构实现上的不同,需要有移植的工作。

2.2 Tracealyzer 分析工具

Tracealyzer 软件工具能够快速、轻松地收集多任务软件有用和有意义的行为,可以快速集成到现有的开发环境,通过快照模式或流模式采集系统运行时数据。Tracealyzer 提供超过 30 种视图可视化 RTOS 运行时行为,视图间以直观的方式相互关联,洞察运行时行为(包括任务运行时间信息、各任务之间的通信流、CPU 的使用率等),帮助开发人员解决问题,提高软件的可靠性,改善软件的性能。

3 FreeRTOS 移植到 GD32VF103 MCU

FreeRTOS 内核绝大部分都采用 C 语言编写,只有与处理器相关的上下文切换采用汇编语言实现,目的是保证上下文切换的效率。将 FreeRTOS 移植到 GD32VF103 MCU 上的关键要点是实现以下 4 个步骤:开启和关闭中断的方式;进入和退出临界区的方式;产生周期性的中断作为系统的时钟节拍;任务的上下文切换。

3.1 中断管理和临界区实现

代码的临界区也称为代码的临界段,这部分代码在执行时不允许被打断。FreeRTOS 的临界区通过关中断来实现,在进入临界段之前须关中断,而临界段代码执行完毕后要立即开中断。GD32VF103 MCU 的 ECLIC 中断控制器有一个中断目标阈值级别寄存器(mth),可以实现部分中断屏蔽,优先级别低于该阈值的中断将不会被响应。在移植 FreeRTOS 时,通过设置 mth 来实现开关中断,对于优先级别比阈值高的中断则不受 FreeRTOS 管理,中断不存在额外的延迟。

3.2 系统时钟节拍支持

操作系统需要一个时钟节拍,以实现系统的延时、超时等与时间相关的处理。时钟节拍是特定的周期性中断,中断的周期就是节拍的时间。节拍的时间长短根据实际应用决定,时钟节拍的频率越高,系统的开销就越大。RISC-V 架构定义了一个 64 位宽度的 mtime 计数器,当 mtime 计数值增加到与 mtimecmp 寄存器预设的值相等时,可以产生中断。选择 mtime 计数器来产生系统时钟节拍,根据 mtime 的时钟频率和系统节拍频率算出 mtimecmp 的值,当中断发生后,通过改写 mtimecmp 或者 mtime 的值来清除中断。

3.3 实现上下文切换

上下文是某一时间点 CPU 的寄存器内容,FreeRTOS 能够正确完成任务调度的关键是上下文切换。上下文切换的过程包括:把即将退出运行态的任务的运行现场保存到其任务堆栈;从下一个要运行的任务的堆栈中恢复它的运行现场。上下文切换的时间应尽可能短,一般由汇编代码编写,作为操作系统移植的一部分。上下文切换分为任务级别及中断级别的切换。上下文切换的代码通常放在异常处理程序中,该异常的优先级别应设置为最低。

在 RISC-V 架构的处理器上,能够用来作为任务切

换的异常有两种：ecall 异常和软件中断。ecall 异常通过调用 ecall 指令来触发；软件中断通过往 msip 寄存器写“1”触发，写“0”清除。GD32VF103 的软件中断连接到 ECLIC 单元进行统一管理。我们实现的 FreeRTOS 移植选用软件中断作为上下文切换的实现机制。

3.4 移植文件修改

FreeRTOS 与处理器相关的移植代码存储在 portmacro.h、portASM.s 和 port.c 三个文件当中。portmacro.h 头文件定义了 FreeRTOS 使用的数据类型、进入和退出临界区的宏、实现开关中断的宏，以及触发和清除软件中断的宏。

portASM.s 中用汇编语言实现 vPortStartFirstTask()函数，用于启动第一个任务，它的核心操作是从 pxCurrentTCB 中取出当前就绪任务中优先级最高任务的堆栈指针 SP，通过 SP 恢复寄存器现场。此外，实现软件中断的服务函数 eclic_msip_handler()将当前的寄存器现场（通用寄存器 x1、x5～x31，机器模式状态寄存器 mstatus 以及机器模式异常 PC 寄存器 mepc）保存到当前在运行任务的堆栈当中，然后从 pxCurrentTCB 取出下一个就绪中优先级最高任务的堆栈指针 SP 恢复寄存器现场，完成任务的上下文切换。

port.c 文件中重点实现堆栈初始化函数 pxPortInitialiseStack()、启动 FreeRTOS 调度器特定的处理函数 xPortStartScheduler()、系统时钟节拍定时器初始化函数 vPortSetupTimerInterrupt()，以及系统时钟节拍中断服务函数 xPortSysTickHandler()。这几个函数分别需要根据 RISC-V 架构和 GD32VF103 MCU 硬件特性来实现其功能。

3.5 移植测试和验证

验证移植采用调试和借助相应的辅助工具来进行，使用 IAR EWRISC-V 建立项目，在代码中创建两个用户任务进行调试，代码调试时需要验证：

① 通过在系统时钟节拍 ISR 和软件中断 ISR 中添加断点，结合 RISC-V 的 mtime 和 mcycle 寄存器验证系统时钟节拍正确产生，且软件中断能够正常触发。

② 启动第一个任务时，通过添加断点查看从任务堆栈中恢复的寄存器内容是否与堆栈初始化时写入的内容一致，从而测试 pxPortInitialiseStack()和 vPortStartFirstTask()函数的工作正确性。

③ 在执行任务上下文切换时，在软件中断服务程序中添加断点，单步执行，同时通过 EWRISC-V 的 memory 观察窗口查看压栈到当前任务堆栈中的内容是否与对应的寄存器内容相同；在恢复下文时，检查从下一个执行任务堆栈中恢复的寄存器内容是否与堆栈中的一致。验证 eclic_msip_handler()软件中断服务函数的上下文切换正确性。

④ 启动第一个任务和任务上下文切换的代码验证能正常工作，移植的 FreeRTOS 已可以实现基本的任务调度，接着再继续测试开关中断操作和临界区是否正常。测试开关中断需要增加另外一个外设中断，将其优先级别分别设置大于或小于 mth 阈值进行测试，代码中手动调用开关中断操作 API，检测中断触发是否与设计的模式一致，验证 FreeRTOS 对中断的控制。采用同样的方法测试进入和退出临界区。

⑤ 通过 FreeRTOS 系统服务调用测试，测试系统的各项服务（如信号量、消息队列、事件标志等）是否正常，并测试在受 FreeRTOS 管理的 ISR 中发信号、消息等操作是否正确。

基础调试测试都通过之后，已经基本可以验证移植是否成功。在此基础上还可以借助额外的工具继续验证，例如通过 EWRISC-V 自带的 FreeRTOS 调试插件显示的信息进行确认（如图 2 所示）。

Tasks

Task Name	Task Number	Priority/actual	Priority/base	Start of Stack	Top of Stack	Min Fr
Led1Task	1	4	4	0x20003b68	0x20003eb8	Off
Led2Task	2	3	3	0x20003ff8	0x20004348	Off
Led3Task	3	2	2	0x20004488	0x200047d8	Off
ButtonTask	4	5	5	0x20004918	0x20004c48	Off
IDLE	5	0	0	0x20004e08	0x20005178	Off

图 2 EWRISC-V RTOS 调试插件-Task 窗口

4 Tracealyzer 分析工具的应用

4.1 移植跟踪记录器库

Tracealyzer 的跟踪记录器库是运行在嵌入式目标端的一个软件库，与 FreeRTOS 项目集成在一起，负责记录 RTOS 在运行时产生的事件。记录的事件如果是存储在 RAM 中，这种工作方式称为快照模式（Snapshot）；如果是通过通信端口实时发送到 PC 端软件，则工作在流模式（Streaming）。跟踪记录器库不依赖于处理器硬件，只需要使用一个高精度的定时器产生时间戳，为记录的事件添加时间信息。

RISC-V 架构的处理器可以利用内核的 mcycle 计数器来产生时间戳，mcycle 是一个 64 位的计数器，对 CPU 的周期进行计数，所以频率与 CPU 时钟相同，精度非常高。mcycle 由两个 32 位的寄存器组成，Tracealyzer 只需要使用低 32 位寄存器。在 trcHardwarePort.h 中定义 GD32VF103 的时间戳实现接口：

```
#define TRC_HWTC_TYPE TRC_FREE_RUNNING_32BIT_INCR
#define TRC_HWTC_COUNT read_csr(mcycle)
#define TRC_HWTC_PERIOD 0
```

```
#define TRC_HWTC_DIVISOR 4
#define TRC_HWTC_FREQ_HZ TRACE_CPU_CLOCK_HZ
#define TRC_IRQ_PRIORITY_ORDER 1
```

Tracealyzer 也需要实现临界区，防止在记录事件时发生任务切换，导致产生错误。实现是通过关中断，可以直接使用 FreeRTOS 的关中断机制在 trcKernelPort.h 中实现临界区的宏接口：

```
#define TRACE_ALLOC_CRITICAL_SECTION() int __irq_status;
#define TRACE_ENTER_CRITICAL_SECTION() __irq_status=portSET_INTERRUPT_MASK_FROM_ISR();
#define TRACE_EXIT_CRITICAL_SECTION() portCLEAR_INTERRUPT_MASK_FROM_ISR(__irq_status);
```

以上是将 Tracealyzer 跟踪记录器库移植到一款处理器上需要做的工作。

4.2 跟踪并分析 FreeRTOS 应用

将 Tracealyzer 跟踪记录器库添加到 EWRISC－V 的 FreeRTOS 项目中，并进行必要的配置，工作在快照模式。在应用代码中创建 4 个任务，分别为 Led1Task～Led3Task、ButtonTask，任务优先级依次递增，程序运行一段时间，将快照数据通过 EWRISC－V 保存成 Hex 文件并加载到 PC 端的 Tracealyzer 中进行分析。

如图 3 所示，通过水平时间轴视图查看各个任务的执行情况：每个任务或中断占一行，从左向右为时间轴的方向，行中有色矩形为该任务或中断的一次执行实例。由时间轴窗口可以快速地预览整个运行过程中系统的执行情况，放大窗口的时间分辨率之后可以仔细了解任务执行时相关的内核事件和时间信息。

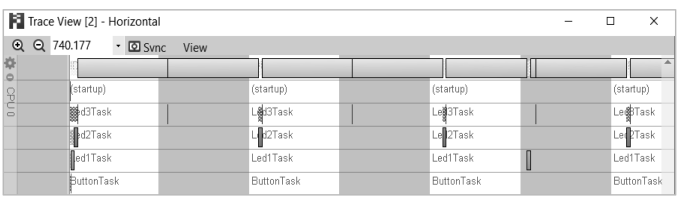

图 3 Tracealyzer 水平时间轴视图

4.3 任务的时间量分析

RTOS 的任务或者中断称为参与者(Actor)，参与者的一次执行称为实例。分析任务的执行通常需要了解如下几种时间：起始和结束时间，参与者实例的开始和结束时间；执行时间，参与者实例使用的 CPU 时间量，不包括抢占；响应时间，从参与者实例开始到结束的时间(更确切地说，任务的响应时间是从任务开始准备就绪时计算的，即内核将任务的调度状态设置为就绪的时间)；等待时间，图 4 所示实例中参与者实际没有执行的时间，计算方式为[(结束时间-开始时间)－(执行时间)]；启动时间，从任务就绪到开始执行之间的这段时间。

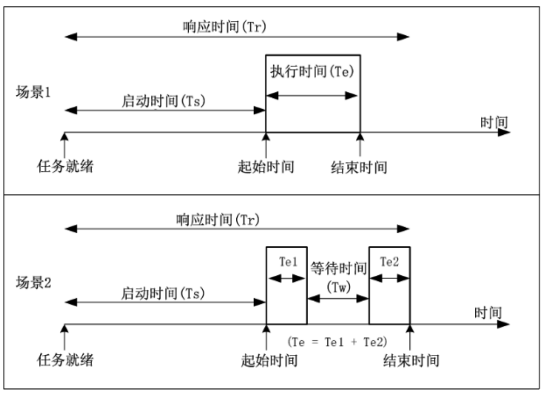

图 4 FreeRTOS 分析基础-实例的时间关系(场景 2 发生了抢占)

以 Led3Task 的第 38 个实例(如图 5 所示)为例，等待时间为从任务就绪到任务代码开始执行的这段时间，时长为 7 μs，因为该实例未发生被抢占的情况，所以这 7 μs 的时间是完成任务上下文切换所需的时间。

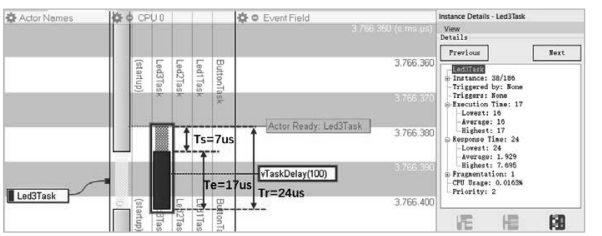

图 5 Led3Task 任务实例 38 的时间量

通过跟踪到的事件和记录的时间戳信息，Tracealyzer 能够生成多种视图来观测系统运行时存在的问题，例如设计缺陷导致的线程饥饿、死锁等，以及发现系统中不必要的延迟，帮助开发者解决系统的问题，提高嵌入式系统的实时性。这些问题使用传统调试手段难以发现，而且效率不高。

5 结 语

本文介绍了基于 RISC－V 指令集的微控制器 GD32VF103 在 IAR EWRISC－V 工具链上移植 Free RTOS 的过程，以及对移植的系统进行验证的方法。在此基础上移植 Tracealyzer 跟踪记录器库，并通过跟踪 FreeRTOS 系统运行进一步观测系统的运行，并对任务的执行进行了分析。嵌入式与物联网是 RISC－V 最活跃的应用市场，RISC－V 给嵌入式系统开发带来许多优势，越来越多的开发者开始尝试 RISC－V。以上工作对于基于 RTOS 技术的 RISC－V 嵌入式开发与应用会有所帮助。

参考文献

[1] 芯来科技. Bumblebee 内核指令架构手册[EB/OL]. [2020－09]. https://www.riscv－mcu.com/quickstart－quickstart－index－u－pdf－id－8.html.

在 DCASE2019 测试集和 UrbanSound8K 测试集上，模型的识别精度如表 1 和表 2 所列，并且与一些最新的方法进行了对比。

表 1　精度比较(DCASE2019)

模　型	测试精度(DCASE)
Audio－Resnet	80.2%
ResNet－like model[14]	76.6%
Baseline system[15]	62.5%

表 2　与其他方法的精度比较

模　型	测试精度(UrbanSound8K)
Audio－Resnet	76.4%
Unsupervised feature learning[16]	73.6%
Baseline system[17]	68%

从表中可以看到，与其他先进的分类方法相比，本文提出的 Audio－Resnet 模型在分类精度上有较大提高。

4　结　语

本文针对声源目标分类中小样本训练时分类模型性能不佳的问题，使用深度学习方法对不同声源发出的声音数据进行分类，使用 log－mel 声谱图特征作为特征预提取方法，采用基于 Resnet 网络结构的分类模型对预提取特征数据进行分类处理，建立了识别效果良好的深度学习声信号分类模型 Audio－Resnet。该模型性能在 DCASE2019 和 UrbanSound8K 数据集上得到了验证，实现了良好的效果，在声源探测领域具有一定的工程应用价值。

参考文献

[1] ISNARD V，CHASTRES V，VIAUD－DELMON I，et al. The time course of auditory recognition measured with rapid sequences of short natural sounds[J]. Scientific Reports，2019，9(1)：8005.

[2] 樊新海，石文雷，张传清. 基于 VMD 多尺度熵和 ABC－SVM 的装甲车辆识别[J]. 装甲兵工程学院学报，2018，32(6)：68－73.

[3] 孙国强，樊新海，石文雷. 基于 MFCC 和支持向量机的装甲车辆识别研究[J]. 国外电子测量技术，2017，36(10)：31－35.

[4] RUBEN GONZALEZ－HERNANDEZ F，PASTOR SANCHEZ－FERNANDEZ L，SUAREZ－GUERRA S，et al. Marine mammal sound classification based on a parallel recognition model and octave analysis[J]. Applied Acoustics，2017(119)：17－28.

[5] SANCHEZ FERNANDEZ L P，SANCHEZ PEREZ L A，CARBAJAL HERNANDEZ J J，et al. Aircraft Classification and Acoustic Impact Estimation Based on Real－Time Take－off Noise Measurements[J]. Neural Processing Letters，2013，38(2)：239－259.

[6] 陈心昭. 噪声源识别技术的进展[J]. 合肥工业大学学报(自然科学版)，2009，32(5)：609－614.

[7] 竺乐庆，张真. 基于 MFCC 和 GMM 的昆虫声音自动识别[J]. 昆虫学报，2012，55(4)：466－471.

[8] 石文雷，樊新海，张传清. 基于频谱动态特征和 CS－SVM 的装甲车辆识别[J]. 计算机应用，2018，38(S1)：44－47，72.

[9] 董语诗，时浏艺，丁一坤. 基于 HMM 的声场景分类[J]. 信息化研究，2018，44(3)：39－47，52.

[10] 中兴通讯采用英特尔 FPGA 在深度学习上取得性能突破[J]. 单片机与嵌入式系统应用，2017，17(3)：88.

[11] 庞悦，赵威，张雅楠，等. 基于深度学习的 LSTM 的交通流量预测[J]. 单片机与嵌入式系统应用，2019，19(3)：72－75.

[12] 王枫，陈小平. CNN 深度学习的验证码识别及 Android 平台移植[J]. 单片机与嵌入式系统应用，2019，19(7)：20－22，73.

[13] PIROTTA E，MERCHANT N D，THOMPSON P M，et al. Quantifying the effect of boat disturbance on bottlenose dolphin foraging activity[J]. Biol Conserv，2015(181)：82－89.

[14] Nguyen T，Pernkopf F. Acoustic Scene Classification with Mismatched Recording Devices Using Mixture of Experts Layer[C]//2019 IEEE International Conference on Multimedia and Expo(ICME)，2019.

[15] Mesaros A，Heittola T，Virtanen T. A multi－device dataset for urban acoustic scene classification[J]. 2018，arXiv：1807.09840.

[16] Salamon J，Jacoby C，Bello J P. A Dataset and Taxonomy for Urban Sound Research[C]//acm International Conference on Multimedia，2014.

[17] SALAMON J，JACOBY C，BELLO J P. A Dataset and Taxonomy for Urban Sound Research[C]//proceedings of the acm International Conference on Multimedia，2014.

王鹏程(硕士研究生)，主要研究方向为信息探测与处理。

(责任编辑：薛士然　收稿日期：2020-07-20)

[2] 胡振波. RISC－V 架构与嵌入式开发快速入门[M]. 北京：人民邮电出版社，2019.

[3] 何小庆. 3 种物联网操作系统分析与比较[J]. 微纳电子与智能制造，2020(3).

[4] DAVID PATTERSON，ANDREW WATERMAN. RISC－V 手册：一本开源指令集的指南，2018.

[5] Using FreeRTOS on RISC－V Microcontrollers[EB/OL]. [2020－09]. https://www.freertos.org/Using－FreeRTOS－on－RISC－V.html.

[6] Jim Cooling. Real－time Operating Systems Book 2－The Practice[M]. Markfield：Lindentree Associates，2017.

(责任编辑：芦潇静　收稿日期：2020-09-10)