计算机科学与技术学院

**毕业设计（论文）开题报告**

题 目：基于Cortex-M3处理器的RTOS设计与实现

姓 名：胡晋玮

学 号：2002020305

班 级：计算机20-4班

专 业：计算机科学与技术

指导教师：朱素霞

2024年 3 月 5 日

**说 明**

1. 正文包括课题研究背景及意义、国内外在该方向的研究现状及分析（文献综述）、研究方案、论文进度计划、参考文献等部分。
2. 研究背景及意义和文献综述部分不少于3000字，报告总体为5000字左右。
3. 参考文献不少于10篇（其中英文文献至少3篇），要有近三年的文献，必须包含学术研究论文，不可全部为图书类文献，文献要在正文中引用。
4. 完成论文的截止日期为毕业学期的6月10日，可按此日期安排论文进度。
5. 正文为小四号字，中文用宋体，英文和数字用Times new roman。
6. 1.5倍行距。
7. 用A4纸双面打印。

**目录**

[1 课题研究背景及意义 1](#_Toc151627861)

[1.1 课题来源（问题提出） 1](#_Toc151627862)

[1.2 选题目的及意义 1](#_Toc151627863)

[2 国内外研究现状及分析（文献综述） 3](#_Toc151627864)

[2.1 VxWorks 3](#_Toc151627865)

[2.2 RT-Thread 3](#_Toc151627866)

[2.3 FreeRTOS 4](#_Toc151627867)

[2.4 其它 5](#_Toc151627868)

[3 论文研究内容 7](#_Toc151627869)

[3.1 研究目标、系统组成和功能、拟解决的关键问题 7](#_Toc151627870)

[3.2 拟采取的研究方法、技术路线、实验方案（含工具、环境）及可行性分析 8](#_Toc151627871)

[4 论文进度计划 11](#_Toc151627872)

[参考文献 12](#_Toc151627873)

# 

# 1 课题研究背景及意义

1.1 课题来源（问题提出）

嵌入式系统是一种将计算机技术、控制技术、通信技术等多种技术融合在一起，为特定的应用目的而设计的专用计算机系统。随着科技的飞速发展，嵌入式系统的应用范围和规模持续扩大，涵盖了工业控制、智能家居、物联网、航空航天、医疗保健等多个领域。为了在有限的资源下提高可靠性、降低开发成本，现代嵌入式系统越来越注重软件模块化和可重用性，其中实时操作系统（RTOS）的角色愈发重要[1]。

微控制器在嵌入式系统中被广泛应用，但由于应用场景的特殊性，系统常常面临资源限制和对实时性、可靠性、安全性等方面的严格要求。为了解决这些问题，使用RTOS是一种常见且高效的方法。RTOS是一种专为实时应用设计的操作系统，具备在规定时间内完成任务、多任务并发执行、任务通信和同步、实时调度算、内存管理等功能，有助于提高系统的性能和效率，同时降低系统的开发和维护成本。

在众多微控制器中，Cortex-M3处理器作为一种低功耗、高性能的处理器，被广泛应用于嵌入式系统[2]。其丰富的外设接口和高效的指令集为嵌入式系统提供了良好的硬件支持，使得基于Cortex-M3处理器的系统在性能和功耗方面表现出色。

然而，尽管Cortex-M3处理器具备卓越的硬件特性，但如何设计一个能够充分利用这些特性，实现高效的任务管理、同步通信、内存管理、性能监控等功能的RTOS，以满足嵌入式系统对实时性、可靠性和功耗效率要求，仍然是一个具有挑战性的任务。

本课题旨在深入研究这一问题，探索在嵌入式系统中如何进行RTOS的设计与实现，以充分发挥Cortex-M3处理器的潜在性能，提高系统的实时性、可靠性和功耗效率。

1.2 选题目的及意义

本课题的研究目的是设计一个基于Cortex-M3处理器的RTOS，实现高效的任务管理、任务间的同步通信、内存管理、性能监控等功能，以满足嵌入式系统对实时性、可靠性和功耗效率要求。本课题将深入研究RTOS的设计与实现，利用Cortex-M3处理器的硬件特性，以提高系统的性能和效率，同时降低开发和维护成本。

本课题的选题背景是当前嵌入式系统的快速发展和广泛应用。随着科技的进步，嵌入式系统在各个领域中扮演着越来越重要的角色，而实时操作系统（RTOS）作为嵌入式系统的核心组成部分，对于系统的可靠性、稳定性和实时性起着至关重要的作用[3]。例如在工业自动化控制系统、医疗设备监测系统等对实时性要求极高的场景中，RTOS的性能直接关系到系统的稳定运行和数据的准确处理。

Cortex-M3处理器作为一种被广泛应用于嵌入式系统的低成本处理器，具有高性能、低功耗和丰富的外设资源等特点。然而，传统的基于Cortex-M3处理器的嵌入式系统往往采用裸机编程的方式，缺乏足够的实时性和可扩展性，存在非常单一的流向、极低的CPU利用率等缺点[2]，针对这些缺点，采用RTOS是一种常见且极为有效的解决方案。

因此，设计和实现一个基于Cortex-M3处理器的RTOS具有重要的意义。

**在对社会的影响方面**，通过设计和实现一个基于Cortex-M3处理器的RTOS，可以提高嵌入式系统的实时性和可靠性。RTOS具备任务调度、优先级管理、内存管理等功能，可以实现多任务并发执行，充分发挥处理器性能，保证各个任务的实时性和响应性[3]。此举有助于提高系统性能，为各个行业提供更高效、更可靠的解决方案，推动社会在工业自动化、医疗监测等领域的技术水平。

**在对环境的影响方面，**嵌入式系统在实际应用中往往涉及到能源管理和环境保护等方面的问题。通过设计和实现一个高效的RTOS，可以优化嵌入式系统的资源利用，降低功耗，减少对环境的负面影响。此外，RTOS还可以提供任务间的通信机制，使得不同任务之间可以高效地共享资源和协同工作，提高系统的效率和性能[3]，从而减少能源消耗和环境污染。因此，本课题的研究对于环境保护和可持续发展具有一定的积极影响。

综述所述，本课题的研究目的是设计并实现一个基于Cortex-M3处理器的RTOS，来更合理、更有效地利用CPU的资源，简化应用软件的设计，缩短系统开发时间，更好地保证系统的实时性和可靠性，同时在社会和环境方面产生积极影响。

# 2 国内外研究现状及分析（文献综述）

* 1. VxWorks

VxWorks是由美国WindRiver公司于1983年推出的一款嵌入式实时操作系统（RTOS），是嵌入式开发环境的重要组成部分。作为全球成熟度最高、应用最广泛的嵌入式实时操作系统之一[4]，其凭借良好的持续发展能力、高性能的内核以及友好的用户开发环境，被广泛运用于医疗基础设施、航空航天、船舶等关键基础设施以及军事领域[5]。

VxWorks操作系统提供了灵活简便的功能组件裁剪方案，使用户能够根据实际需求轻松配置个性化的操作系统内核。默认的VxWorks系统内核非常小，仅约8KB，功能有限。然而，功能强大的VxWorks支持大量可裁剪的组件，包括系统文性系统、网络服务、多任务、多线程支持、图形显示控制、远程服务、域名解析服务、文件传输服务、外设支持等高级功能组件。

用户可以借助WindRiver公司提供的软件集成开发工具Wrkbench等，通过创建内核映像项目，利用图形界面轻松配置和定制所需的系统内核。这种高度的灵活性允许用户根据实际应用需求快速定制或二次开发符合其需求的操作系统镜像。这不仅为软件开发提供了灵活性和便利性，还有助于节约磁盘空间、去除不必要的功能模块，提高系统运行速度和性能。

在实时性方面，由于VxWorks操作系统的高度裁剪性，系统资源占用最小，开销可控。系统设计初期即以实时性为设计标准，采用任务轮转和任务优先级相结合的调度算法，确保在相同运算性能的计算机上具有更高的实时性和可靠性。系统的多任务并发设计更符合真实世界的并发需求，且采用的调度算法有助于保持较小的时间开销，提升系统的运行速度和性能[5]。

VxWorks的主要缺点在于其昂贵的价格以及与之相关的高昂的软件开发和维护成本。通常，VxWorks只提供二进制代码，而不提供源代码，这限制了用户对系统的深度定制和调整[6][7]。此外，VxWorks支持的硬件种类有限，这可能对一些项目的硬件选择带来限制。

* 1. RT-Thread

RT-Thread，全称是 Real Time-Thread，诞生于2006年，是国内以开源中立、 社区化发展起来的一款组件丰富、易扩展、低功耗、安全性高的国产高可靠实时操作系统，由睿赛德科技负责开发维护和运营。

RT-Thread是一个抢占式实时多任务操作系统，采用面向对象的设计方法，结构清晰，易于裁剪和移植。RT-Thread不仅是一个实时内核，而且是一个丰富的中间层组件。如下图1所示，该系统采用分层结构，从下到上依次为：CPU体系结构、内核层（Kernel）、组件和服务层（Components）、软件包（Packages）。实时内核是系统的核心，包括线程管理、时钟管理、内存管理、信号量、事件、消息队列等。任务线程调度是灵活的，具有多种内存管理算法，并支持多种同步和通信机制。组件和服务层提供了完整的中间件组件，如文件系统和图形库。软件包丰富多样，包括60多个物联网、系统、外设库和驱动程序[8]。

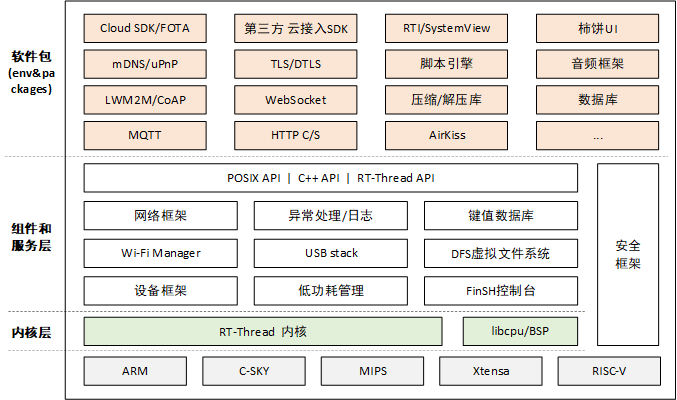


图1 RT-Thread 系统结构

RT-Thread 不依赖于特定的内存分配方式，而是通过对象容器中内核对象动态与静态相结合的方法，来显著提高系统的灵活性。凭借稳定丰富的系统功能，RT-Thread在新能源、航空和医疗等领域被广泛应用，其高度可靠的实时性得到了证实[9]。

* 1. FreeRTOS

FreeRTOS由美国的Richard Barry于2003年发布，是一个轻量级的嵌入式实时操作系统，采用C语言开发。该操作系统支持无限数量的任务，并提供优先级调度算法。每个任务都被分配一个优先级，处理器会优先执行处于就绪态的优先级最高的任务。此外，FreeRTOS还支持轮转调度算法，允许不同任务共享相同的优先级，并且同一优先级的任务可以共享处理器的使用时间[10]。

作为迷你实时操作系统内核，FreeRTOS功能丰富，包括任务管理、时间管理、信号量、消息队列、内存管理、记录功能、软件定时器、协程等。这使得它可以满足较小系统的需求。

由于实时操作系统需要占用一定的系统资源（尤其是RAM资源），只有少数实时操作系统如μC/OS-II、embOS、salvo、FreeRTOS能够在小RAM单片机上运行。相比商业操作系统如μC/OS-II、embOS等，FreeRTOS操作系统是完全免费的，具有源码公开、可移植、可裁减、调度策略灵活等特点。这使得它可以方便地移植到各种单片机上运行[11]。

相对于常见的μC/OS-II操作系统，FreeRTOS操作系统也有一些不足之处。其不足之处。一方面体现在系统的服务功能上，如FreeRTOS只提供了消息队列和信号量的实现，无法以后进先出的顺序向消息队列发送消息；另一方面，FreeRTOS只是一个操作系统内核，需外扩第三方的GUI(图形用户界面)、TCP/IP协议栈、FS(文件系统)等才能实现一个较复杂的系统，不像μC/OS-II可以和μC/GUI、μC/FS、μC/TCP-IP等无缝结合。

* 1. 其它

除上述三款主流实时操作系统外，国内外还有一些先进的RTOS。

RT-Linux是由美国新墨西哥州的fsmlabs实验室基于Linux开发的嵌入式实时操作系统[12]。该操作系统部分支持POSIX.1b标准，其内核设计巧妙，对Linux内核进行了最小化的修改。RT-Linux充分利用了Linux下丰富的软件资源，使其在硬实时和软实时情况下均可使用。

WindowsCE为32位，属微软公司旗下，是一种紧凑、高效、可伸缩的操作系统,具有多线程、完整优先级、多任务、完全抢占的特点[13]。WindowsCE（Wince）相对简易开发，周期短且内核完善，可灵活支持通信[14]，GUI丰富且开发功能完善，能更大程度地控制调度机制。WinCE的缺点之一是昂贵的版权费用，这是不可避免的成本。此外，WinCE没有充分考虑实时应用，难以在多种硬件平台上实现良好的支持。其定制性较差，且对系统资源的需求较高。在网络功能方面，WinCE应用相对较少[6]。

μC/OS是由Micrium公司创始人Jean J. Labrosse编写的嵌入式实时操作系统，经过三个版本的演进：μC/OS、μC/OS-II、μC/OS-III[15]。这一系列操作系统在嵌入式实时领域得到广泛应用。μC/OS-III不仅延续了前两代的实时多任务内核，还引入了丰富的软件包，包括文件系统、图形用户界面(GUI)、网络协议栈等，同时在任务调度、中断处理、时钟节拍、哈希散列表等方面进行了改进，使其更适用于复杂的嵌入式系统设计[11]。

除了国外先进的嵌入式实时操作系统，国产系统也在不断发展，包括DJYOS、Alios Things、Huawei LiteOS、SylixOS等。DJYOS以其卓越的实时性能、快速的中断响应速度和基于事件的核心调度而脱颖而出，主要应用于光伏发电、新能源和电力控制系统等领域。Alios Things具有极简的开发特性，与云端紧密集成，拥有丰富的组件和对多环境的广泛支持，主要应用于智能办公、智能家居和智慧城市等场景。Huawei LiteOS则以其轻量级、低功耗、互联互通、丰富的组件和快速开发的特点而著称，主要应用于车联网、穿戴设备和智能家居等领域。SylixOS具备全面的功能、稳定可靠、易于开发、高可靠性和高实时性等特征，被广泛应用于车载控制平台、核电控制系统、固定翼无人机、物联网通讯卫星等多个场景[6][11]。

# 3 论文研究内容

3.1 研究目标、系统组成和功能、拟解决的关键问题

**（1）研究目标**

研究并分析Cortex-M3处理器的体系结构特性以及RTOS的设计原理和实现方法，设计并实现一个适用于Cortex-M3处理器的嵌入式实时操作系统（RTOS）。

**（2）系统组成和功能**

预计设计的RTOS的系统结构如图2所示：

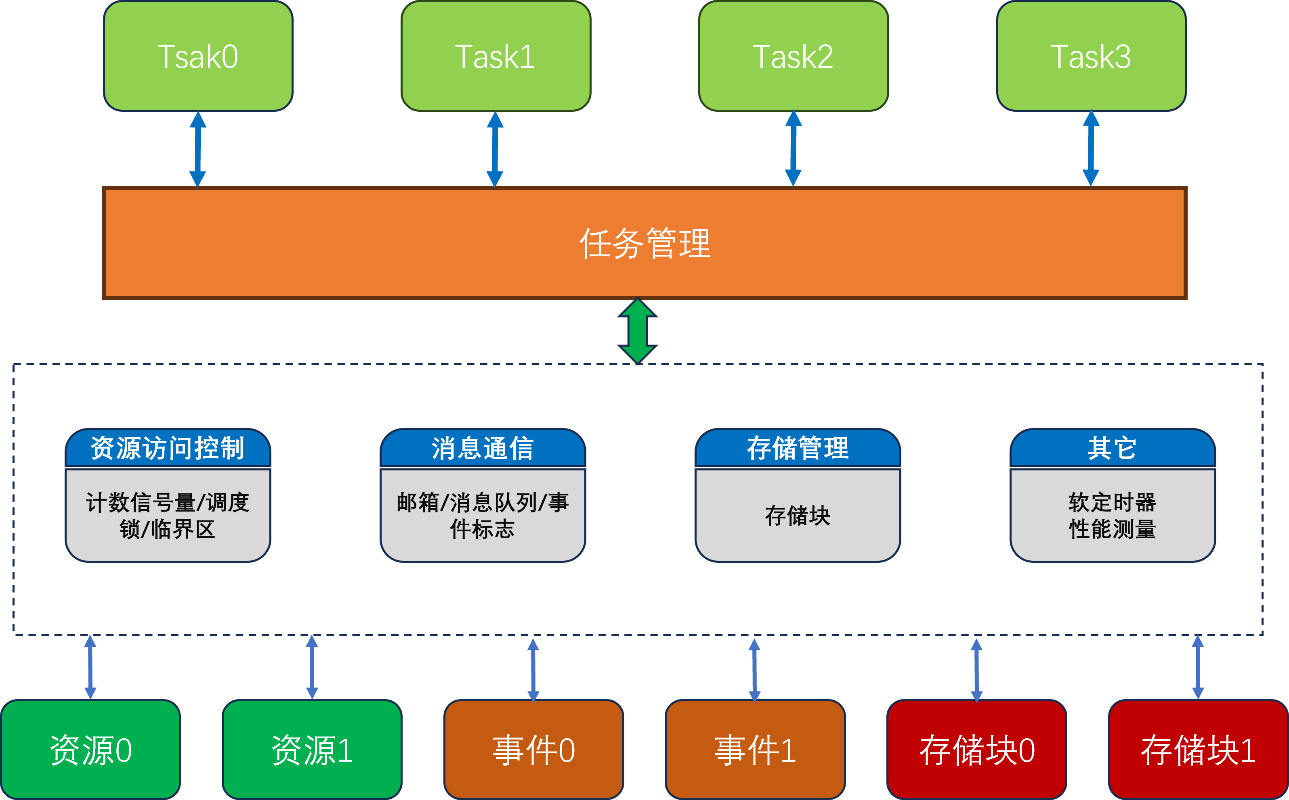


图2 本课题的RTOS拟定的系统结构

任务管理模块：该模块实现任务调度器，负责任务的创建、删除和切换。采用抢占式调度算法，支持最多32个优先级，同一优先级的任务按时间片轮转调度，确保任务能够按照优先级和时间片得到合理调度。系统不限制任务数量。

资源管理模块：提供计数信号量，用于实现任务的同步和互斥。提供互斥信号量，用于确保多任务对共享资源的互斥访问。

消息通信模块：支持邮箱机制，用以实现多个任务之间的消息通信。支持消息队列，用于任务之间的有序通信，包括消息的排队和处理。支持事件标志，实现事件标志检测与等待，以便任务可以等待或检测某个事件的发生。

存储管理模块：提供基于块的存储管理，以确保任务可以安全地分配和释放内存块。

其它模块：提供软定时器，允许任务在指定的时间间隔内执行特定的操作或触发相应事件。提供任务堆栈检测功能，用于检测堆栈的溢出情况以及诊断任务堆栈的使用情况。提供CPU利用率检测功能，用于监测系统的性能和效率。

**（3）拟解决的关键问题**

1. 如何确保设计实现的RTOS能够高效地适配Cortex-M3处理器的架构，以充分发挥其性能和特性？

2. 如何实现抢占式调度和时间片轮转调度，以确保任务按照优先级和时间片得到合理调度？

3. 如何有效管理任务的创建和删除，以保障系统的稳定性和灵活性？

4. 如何确保互斥信号量和计数信号量的正确实现，以防止竞态条件和资源争用？

5. 如何设计基于块的存储管理，以确保任务可以安全地分配和释放内存块？

6. 如何实现有效的邮箱通信机制，使任务能够异步地通过邮箱传递消息？

7. 如何设计消息队列，以支持任务之间的有序通信，确保消息的正确排队和处理？

8. 如何实现事件标志检测与等待，以便任务可以等待或检测某个事件的发生？

9. 如何实现任务堆栈检测功能，以监测任务的堆栈溢出情况？

10. 如何设计功能强大的CPU利用率测量功能，以监测系统的性能和效率？

3.2 拟采取的研究方法、技术路线、实验方案（含工具、环境）及可行性分析

**（1）拟采取的研究方法**

本研究将采用综合性的研究方法，包括文献综述、理论分析、模块化设计、仿真验证和实际硬件实验。首先，通过文献综述，深入了解Cortex-M3处理器的特性、RTOS的设计原理以及相关领域的最新研究成果。其次，进行理论分析，明确RTOS的设计目标和系统功能，提取关键问题，并初步确定解决方案。接下来，采用模块化设计方法，将RTOS划分为不同的功能模块，并详细设计每个模块的接口和实现方式。最后，通过仿真验证和实际硬件实验，验证设计的RTOS在Cortex-M3处理器上的适配性和性能。

**（2）技术路线**

技术路线主要包括以下几个关键步骤：

a. 理论分析和文献综述：深入了解Cortex-M3处理器和RTOS的设计原理，明确研究目标和系统功能。

b. 模块化设计：将RTOS划分为任务管理、资源管理、消息通信、存储管理等功能模块，并设计各模块的接口和实现方式。

c. 编码实现：使用C语言和汇编语言，根据设计的模块进行具体编码实现。

d. 仿真验证：利用仿真工具，如Keil中集成的Cortex-M3芯片模拟器，对设计的RTOS进行功能验证和性能评估。

e. 实际硬件实验：将设计的RTOS移植到Cortex-M3处理器的开发板上进行实际硬件实验，验证系统的可靠性和性能。

**（3）实验方案**

为实现研究目标，采用以下工具与环境：

开发工具： 使用集成开发环境（IDE）Keil MDK，以提高RTOS的开发、编译和调试效率。

仿真工具： 采用Keil内置的仿真工具，模拟Cortex-M3处理器的运行环境，进行RTOS的功能验证和性能评估。这能够在开发早期快速验证设计，并减少对实际硬件的依赖。

实际硬件环境： 选择支持Cortex-M3处理器的开发板，例如STM32系列开发板。通过在实际硬件上移植RTOS，进行系统的真实性能测试。

实验方案包括仿真验证和实际硬件实验两个阶段。在仿真验证阶段，使用Keil内置的模拟器等仿真工具，搭建Cortex-M3处理器的仿真环境，加载设计的RTOS，并进行功能验证和性能评估。在实际硬件实验阶段，将设计的RTOS移植到Cortex-M3处理器的开发板上，通过实际硬件测试验证系统在实际环境中的可行性和性能表现。

**（4）可行性分析**

通过技术、经济和时间三个方面的考虑，对研究的可行性进行分析：

* **技术可行性：** Cortex-M3处理器和RTOS设计原理已经有深入研究，相关技术实现是可行的。开发工具和仿真工具的广泛应用也支持系统的快速开发和验证。
* **经济可行性：** 选择常用的开发工具和开发板，成本相对较低。这降低了项目的经济负担，使得研究在可控的成本范围内进行。
* **时间可行性：** 通过合理的工作计划和任务划分，能够确保研究按时完成。采用仿真工具在早期进行验证，可以在研究的各个阶段更好地掌握进度。

综合考虑上述因素，本研究在技术、经济和时间上均具有可行性。

# 4 论文进度计划

论文各环节的进度安排如表1所示。

|  |
| --- |
|  |

表1 进度计划表

|  |  |
| --- | --- |
| 时间安排 | 实施进度 |
| 2023.10.23~2023.12.31 | 了解毕业设计内容，查阅资料，阅读大量文献综述。 |
| 2024.01.01~2024.02.29 | 学习并掌握相关技术，了解RTOS的设计与实现原理，Cortex-M3处理器体系结构。 |
| 2024.03.01~2024.04.30 | 编写代码实现系统功能并进行验证。 |
| 2024.05.01~2024.06.10 | 撰写论文，准备答辩。 |

# 参考文献

1. Karnakanti, S. A Survey on Different Real Time Operating Systems[J]. International Journal of Engineering and Advanced Technology (IJEAT), 2021, 10(5): 221-223.
2. 郑尧尹. 基于Cortex-M3核的MCU的设计及应用[D].电子科技大学,2022.
3. Ronak Jain, Avani Pareek, Brakha Shrivastav, Rekha Vijay. A Review on RTOS and its Applications[J]. Advancement and Research in Instrumentation Engineering, 2023, 6(1).
4. HAMBARDE P, VARMA R, JHA S. The Survey of Real Time Operating System: RTOS[C/OL]//2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies, Nagpur, India. 2014.
5. 张国超,金巧园,代中华.基于VxWorks的视频记录重演系统设计与实现[J].电子技术与软件工程,2022(11):100-104.
6. 易家颖,司宾强,朱纪洪. 嵌入式实时操作系统浅析[C]//中国高科技产业化研究会智能信息处理产业化分会.第十六届全国信号和智能信息处理与应用学术会议论文集.2022:7.
7. IBRAHIM B R, KHALIFA F M, ZEEBAREE S R M, et al. Embedded System for Eye Blink Detection Using Machine Learning Technique[C/OL]//2021 1st Babylon International Conference on Information Technology and Science (BICITS), Babil, Iraq. 2021.
8. YIMING D, KUI L. Design and Implementation of Brushless DC Motor Control Scheme Based on RT-Thread Operating System[C/OL]//2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), Nanchang, China. 2021.
9. 赵剑川,赵静,欧启标等.实时操作系统RT-Thread启动流程剖析[J].单片机与嵌入式系统应用,2022,22(06):22-25.
10. 范亚南,高军科,韩文斌等.基于FreeRTOS的多连接TCP Server设计[J].仪表技术与传感器,2023(08):96-100.
11. 余小桐. RT-Thread平台的可信启动研究与实现[D].中北大学,2023.
12. González-Baldovinos DL, Guevara-López P, Cano-Rosas JL, Valdez-Martínez JS, López-Chau A. Response Times Reconstructor Based on Mathematical Expectation Quotient for a High Priority Task over RT-Linux. Mathematics. 2022; 10(1):134.
13. 陈瑞杰.实时操作系统RTOS发展概述[J].通信电源技术,2016,33(04):199-200.
14. 瞿伟, 余飞鸿. 基于多核处理器的非对称嵌入式系统 研究综述[J]. 计算机科学, 2021, 48(S1): 538-542.
15. Lian Li, Xianchen Shi. μC/OS-II Based RTOS Kernel[M]. 2019.