Deep Shinglot

CSE-5354 Real-time Operating Systems - Fall 2024 Project

University of Texas at Arlington

Introduction

The Real-Time Operating System (RTOS) project aims to implement a robust preemptive RTOS for the ARM Cortex-M4F microcontroller. This project demonstrates an in-depth understanding of real-time systems concepts and the ability to design and implement features essential for multitasking in embedded environments. The system includes core functionalities such as thread prioritization, scheduling, context switching, and kernel operations, enabling efficient task management and resource sharing.

The implemented RTOS incorporates advanced features like mutexes, semaphores, and sleep functionality, supporting thread synchronization and timing control. It also includes a shell interface for user interaction, providing commands to monitor system resources, manage threads, and configure the kernel. Additionally, preemption and memory protection were integrated to enhance the system's reliability and safety.

Implementation

The RTOS implementation includes the PendSV interrupt service routine, mutex operations, and semaphore management.

1. PendSV Interrupt Service Routine:

   This routine manages task switching by saving the state of the currently running task and restoring the state of the next scheduled task. It also calculates CPU usage for profiling by monitoring system timers and handles memory protection faults, ensuring tasks are safely terminated if they violate memory bounds.

2. Mutex Operations:

    - Lock: When a task requests a mutex, it checks if the resource is available. If it is, the mutex is locked and assigned to the task; otherwise, the task is placed in a waiting queue, optionally adjusting priorities if priority inheritance is enabled.
    - Unlock: A task holding a mutex can release it, granting access to the next waiting task if any—unprotected access to a mutex results in the task being terminated for safety.
3. Semaphore Operations:
    - Wait: Tasks request access to a shared resource by decreasing the semaphore count. The task is queued and blocked if the resource is unavailable until the semaphore is free.
    - Post: A task signals the release of a resource by incrementing the semaphore count and waking the next waiting task if one exists.

Each component ensures proper synchronization and resource allocation across tasks, maintaining the stability and efficiency of the RTOS.

Memory Protection Unit (MPU) configuration has regions with adjusted Subregion Disable (SRD) values, which impact the active subregions in each area. This configuration ensures that access control is maintained for unprivileged mode (user mode), which utilizes the Process Stack Pointer (PSP), in contrast to thread mode, which uses the Main Stack Pointer (MSP). In this project, all tasks are configured to operate in unprivileged mode on their own PSP preventing unauthorized accesses. Interrupt Service Routines (ISRs) are privileged by default, granting them full access rights, this falls under the OS region.

1. Background Region:

- Access: No read, write, and execute (-r-w-x) to user mode.

- SRD: 0x00 (all subregions are active), Coverage 0x0000.0000 0xFFFF.FFFF.
- This serves as the default fallback region with no disabled subregions.

2. Flash Region:

- Access: Given read, write, and execute (+r+w+x) to user mode.
- SRD: 0x00 (all subregions are active), Coverage 0x0000.0000 - 0x0004.0000.
- Flash memory remains fully accessible and executable, suitable for storing application code.

3. OS Region:

- Access: No read, write, and execute (-r-w-x) to user mode.
- SRD: 0xFF (all subregions are disabled), Coverage 0x2000.0000 - 0x2000.0FFF.
- This indicates a highly restricted region where all subregions are disabled, limiting access to safeguard sensitive OS data.

4. Peripheral Region:

- Access: Given read, write, and execute (+r+w+x) to user mode.
- SRD: 0x00 (all subregions are active), Coverage 0x4000.0000 - 0x5FFF.FFFF.
- All subregions are active, ensuring complete accessibility to memory-mapped I/O operations.

5. SRAM Region:

- Access: Given read, write, and execute (+r+w+x) to user mode.
- SRD: 0x00 (all subregions are active), Coverage 0x2000.1000 - 0x2000.7FFF.
- The memory remains fully accessible for task general-purpose use, including data and stack operations.

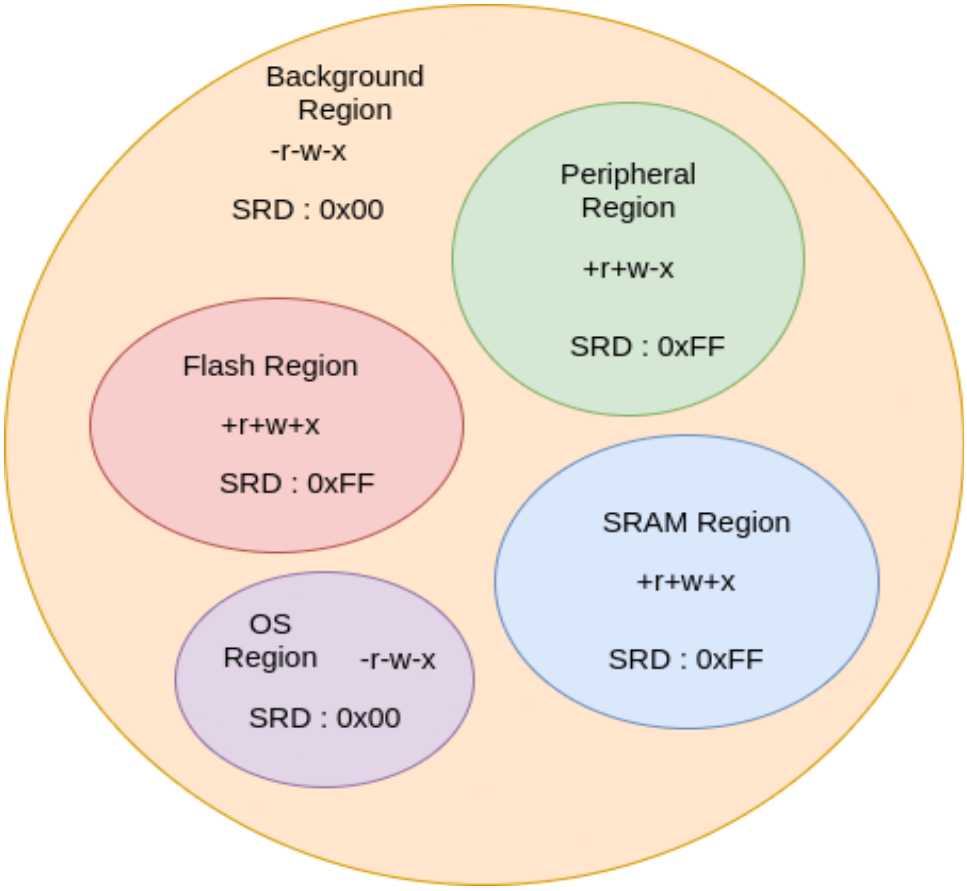

Figure 1: MPU Setup Respect to User Mode

Conclusion

The RTOS implementation for the ARM Cortex-M4F microcontroller showcases a solid grasp of real-time embedded systems, delivering efficient task management through features like thread scheduling, mutexes, semaphores, and context switching. The use of the Memory Protection Unit (MPU) ensures system security by enforcing access controls, with tasks running in unprivileged mode and ISRs operating with full privileges.

This project highlights the ability to design a reliable, scalable RTOS that balances functionality, safety, and efficiency, making it well-suited for multitasking in embedded environments.