Real-Time Operating Systems for Embedded Systems
Final Assignment : RTOS Design

SEMPERE Tao

M.Jasdeep Singh

January 15

Contents

Introduction

Embedded systems are integral to modern technology, playing a crucial role in sectors like automotive control and industrial automation. Their ability to swiftly respond to events with precise timing necessitates a specialized Real-Time Operating System (RTOS). This project addresses the need for an RTOS tailored to real-time embedded systems, emphasizing not only task design but also Worst-Case Execution Time (WCET) analysis for system reliability.

Integrated with FreeRTOS, an open-source RTOS, the project demonstrates practical implementation within the embedded domain. The focus extends beyond theory, incorporating tangible integration into the FreeRTOS framework. Through systematic task definition, WCET analysis, and integration, the project contributes to understanding real-time OS intricacies.

Subsequent report sections detail methodologies, schedulability analysis results, and implementation specifics. This project aims to contribute significantly to the discourse on real-time systems, particularly in embedded applications. By showcasing the practical implications of RTOS implementation, it provides insights for future developments. The project repository, containing the codebase and a detailed report, can be accessed through the provided Git link.

1 Methods

1.1 Tasks, WCET & Period

Determining the Worst Case Execution Time (WCET) is crucial in real-time system design, representing the maximum expected execution time under unfavorable conditions. A practical method involves instrumenting source code with timestamps for critical sections, utilizing temporal functions for precise measurement.

Task period definition in a real-time system is equally critical. Unlike WCET, which may require detailed code analysis, period determination often considers the intrinsic nature of tasks. Tasks dictate their execution frequency, and customer preferences or dependencies between tasks influence period adjustment.

In essence, while WCET determination requires precise code analysis and timestamps, task period definition relies on intrinsic task nature, customer preferences, dependencies, and system flexibility. Both aspects are vital for ensuring an effective real-time system.

Periodic Task 1 (Print ”Working”)

- WCET: 0.039 millisecond.
- Period : 500 milliseconds.
- Motivation: In order to have not too many messages printed, we choose a period of 500 milliseconds, and it will be the same motivation for every periods. We just chose diffierient periods to make sure it works.

Periodic Task 2 (Convert Fahrenheit to Celsius)

- WCET: 0.035 millisecond.
- Period : 1000 milliseconds.

Periodic Task 3 (Multiply two large integers

- WCET: 0.031 millisecond.
- Period : 1500 milliseconds.

Periodic Task 4 (Binary search)

- WCET: 0.040 milliseconds.
- Period : 2000 milliseconds.

Analysis method Fixed Priority (FP) scheduling is a commonly used approach in real-time systems where tasks are assigned priorities based on their periods. The Rate Monotonic (RM) analysis is a specific case of FP scheduling, and it helps determine the schedulability of a set of periodic tasks.

Calculate the utilization (U) of each task by dividing its period (T_i) by its worst-case execution time (C_i) using the formula:

$$U_i = \frac{T_i}{C_i}$$

Sum up the individual utilizations to obtain the total system utilization:

$$U_{\text{total}} = U_1 + U_2 + \ldots + U_n$$

Use the Rate Monotonic (RM) criterion for FP scheduling. The schedulability test is given by the inequality:

$$U_{\text{total}} \leq n(2^{1/n} - 1)$$

where n is the number of tasks.

Compare the total system utilization (U_{total}) with the schedulability threshold. If U_{total} is less than or equal to the threshold, the task set is schedulable. If it exceeds the threshold, further adjustments to task parameters or scheduling policies may be required.

2 Results

We Schedule the tasks in Fixed Priority using FreeRTOS.

Let's see if the task is schedulable.

- Utilization (U_1): $\frac{0.039}{500} = 7.8 \times 10^{-5}$
- Utilization (U_2): $\frac{0.035}{1000} = 3.5 \times 10^{-5}$
- Utilization (U_3): $\frac{0.031}{1500} = 2.07 \times 10^{-5}$
- Utilization (U_4): $\frac{0.040}{2000} = 2 \times 10^{-5}$

$$U_{\text{total}} = U_1 + U_2 + U_3 + U_4 = 7.8 \times 10^{-5} + 3.5 \times 10^{-5} + 2.07 \times 10^{-5} + 2 \times 10^{-5}$$

The schedulability test for Fixed Priority (FP) scheduling is:

$$U_{\text{total}} \leq n(2^{1/n} - 1)$$

For $n = 4$:

$$7.8 \times 10^{-5} + 3.5 \times 10^{-5} + 2.07 \times 10^{-5} + 2 \times 10^{-5} \leq 4(2^{1/4} - 1)$$

$$0.000135 \leq 0.7568$$

This analysis suggests that the updated periodic task set is likely to be schedulable.

Conclusion

This project designed a Real-Time Operating System (RTOS) for embedded systems, focusing on task management, Worst-Case Execution Time (WCET) analysis, and FreeRTOS integration.

WCET was determined through source code instrumentation, and task periods were set based on task characteristics, customer preferences, dependencies, and system flexibility.

Schedulability analysis, using Fixed Priority and Rate Monotonic criteria, showed that adjusted task parameters maintained system schedulability.

Implementation demonstrated the adaptability of the proposed RTOS with FreeRTOS, highlighting its efficiency in addressing real-time constraints. The systematic approach provides insights for designing RTOS in embedded applications.

This project contributes to advancing real-time embedded systems, offering practical solutions for task scheduling and performance analysis. The insights gained can guide the development of robust RTOS solutions for diverse embedded applications.