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

FORNER Jessica

M.Jasdeep Singh

15/01/2024

Abstract

Real-Time Operating Systems (RTOS) are indispensable components in the landscape of embedded systems, particularly when stringent timing requirements and deterministic behavior are essential. This project delves into the intricacies of designing and implementing an RTOS tailored explicitly for embedded environments. The focal points of the endeavor encompass the creation and management of both periodic and aperiodic tasks, comprehensive analysis of worst-case execution times (WCET), meticulous scheduling considerations, and seamless integration with the FreeRTOS platform.

The objective is to provide a comprehensive framework that not only meets the real-time constraints demanded by embedded applications but also demonstrates the adaptability and efficiency of an RTOS in addressing the unique challenges posed by these environments. Through a methodical exploration of task design, WCET analysis, and FreeRTOS integration, this project aims to contribute insights and practical solutions to the complex landscape of real-time embedded systems.

Contents

Introduction

Embedded systems operating in real-time scenarios are omnipresent in contemporary technological ecosystems, ranging from automotive control units to industrial automation. The hallmark of these systems lies in their ability to respond promptly to events and meet stringent timing requirements. An RTOS tailored for such environments becomes pivotal, as it ensures deterministic behavior and enables seamless coordination of tasks with diverse timing constraints.

This project is motivated by the need for a specialized RTOS that can address the intricacies of real-time embedded systems. The challenges encompass not only the design of tasks that adhere to strict timing requirements but also the analysis of worst-case execution times to guarantee system reliability. By integrating with FreeRTOS, a robust open-source real-time operating system, this project aspires to showcase the practical implementation and effectiveness of an RTOS in the embedded domain.

Through a systematic approach to task definition, WCET analysis, and FreeRTOS integration, this project endeavors to provide a comprehensive understanding of the intricacies involved in designing a real-time operating system for embedded environments. The subsequent sections will delve into the methodological aspects, results of schedulability analysis, and the implementation details, ultimately contributing to the broader discourse on real-time systems for embedded applications.

1 Methods

1.1 Tasks, WCET & Period

The determination of Worst Case Execution Time (WCET) is a crucial step in the design of a real-time system. WCET represents the maximum expected execution time for a given task under unfavorable conditions. This measure is essential to ensure that the system meets the real-time constraints imposed by its environment.
A practical method for estimating WCET involves the instrumentation of the source code for each task. This approach entails integrating timestamps at the starting and ending points of critical code sections. These markers enable the precise measurement of time elapsed during the execution of these sections.
Temporal functions from the operating system or programming language are utilized to measure execution time. These functions provide sufficient granularity to capture details of critical code portions.

Determining task periods in a real-time system is a critical step to ensure compliance with temporal constraints. Unlike determining WCET, which may require a detailed code analysis, defining periods can often be based on the intrinsic nature of the tasks.
The nature of the task itself can often dictate how frequently it needs to be executed. Consider the example of making coffee. If this task is frequently requested, the period between executions should be relatively short to quickly respond to customer demands.
Periods can also be determined based on customer requirements. If a customer prefers to receive their coffee shortly after placing an order, the period between executions of the ”make coffee” task should be adjusted accordingly. This can be compared to how often the customer prefers to receive their coffee, whether every 2 minutes, every hour, or another interval.
Dependencies between tasks can also influence period determination. If the ”make coffee” task depends on another task, such as ”heat water,” the period may be adjusted based on how frequently the dependent task needs to be executed.
Period determination should also consider the flexibility of the system. If the system needs to be able to respond to unforeseen requests or adapt to changes in the environment, periods can be set to ensure this flexibility.

Let’s see our choices for the different tasks

1.1.1 Periodic Task 1 (Print "Working")

- WCET: 15×10^{-4} millisecond.
- Period : 15×10^{-3} milliseconds.
- Motivation: The task simply involves printing a message, and the WCET and the period is measured accordingly.

1.1.2 Periodic Task 2 (Convert Fahrenheit to Celsius)

- WCET: 35×10^{-4} millisecond.
- Period : 500 milliseconds.
- Motivation: Temperature conversion is a simple operation, but the WCET includes margins for fluctuations and period is estimated accordingly.

1.1.3 Periodic Task 3 (Multiply two large integers)

- WCET: 20×10^{-4} millisecond.
- Period : 1000 milliseconds.
- Motivation: Multiplying large integers can be a computationally intensive operation, and the WCET and period considers worst-case scenarios.

1.1.4 Periodic Task 4 (Binary search)

- WCET: 25×10^{-4} milliseconds.
- Period : 100 milliseconds.
- Motivation: Binary search on a fixed list is relatively efficient, but the WCET accounts for worst-case situations and as it is more efficient taht a search "one-by-one" list, the period is chosen accordingly.

1.2 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 taks is schedulable.

Periodic Task 1 (Print "Working"):

- WCET: 15×10^{-4} milliseconds.
- Period: 200 milliseconds.
- Utilization (U_1): $\frac{15\times10^{-4}}{200} = 7.5 \times 10^{-7}$

Periodic Task 2 (Convert Fahrenheit to Celsius):

- WCET: 35×10^{-4} milliseconds.
- Period: 500 milliseconds.
- Utilization (U_2): $\frac{35\times10^{-4}}{500} = 7 \times 10^{-7}$

Periodic Task 3 (Multiply two large integers):

- WCET: 20×10^{-4} milliseconds.
- Period: 1000 milliseconds.
- Utilization (U_3): $\frac{20\times10^{-4}}{1000} = 2 \times 10^{-7}$

Periodic Task 4 (Binary search):

- WCET: 25×10^{-4} milliseconds.
- Period: 100 milliseconds.
- Utilization (U_4): $\frac{25\times10^{-4}}{100} = 2.5 \times 10^{-6}$

Now, sum up the individual utilizations to obtain the total system utilization (U_{total}):

$$U_{\text{total}} = U_1 + U_2 + U_3 + U_4 = 0.1 + 7 \times 10^{-7} + 2 \times 10^{-7} + 2.5 \times 10^{-6}$$

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

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

For $n = 4$:

$$0.1 + 7 \times 10^{-7} + 2 \times 10^{-7} + 2.5 \times 10^{-6} \leq 4(2^{1/4} - 1)$$

$$0.10000075 \leq 0.7568$$

This analysis suggests that the updated periodic task set is likely to be schedulable under Fixed Priority scheduling with the chosen periods and worst-case execution times. Adjustments may be made if needed for further validation.

Conclusion

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

WCET determination was crucial, involving source code instrumentation. Task periods were based on task nature, customer preferences, dependencies, and system flexibility.

Schedulability analysis, using Fixed Priority and Rate Monotonic criteria, indicated that the system with adjusted task parameters remained schedulable.

Implementation demonstrated the adaptability of the proposed RTOS with FreeRTOS, showcasing its efficiency in addressing real-time constraints. The systematic approach provides valuable 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 tailored for diverse embedded applications.