Final assigment

Bourdeau Clément & Collin Sam

Course : Real-Time Embedded Systems

Teacher : Jasdeep SINGH

Year : 2024

Abstract

This project presents the development of a Real-Time Operating System (RTOS) tailored to manage a set of specific tasks efficiently. Our RTOS is designed to handle periodic and aperiodic functions with precision, ensuring that each task is executed within its defined time frame. In the sections that follow, we will detail the method used to establish our RTOS, from its fundamental architecture to the implementation of the tasks.

The report will further discuss how we calculate the execution times and validate our system through a series of tests. Our aim is to create an RTOS that is both simple in design and effective in performance, capable of serving as a reliable tool for task management in real-time applications.

You'll find all the files you need to run the project in the GIT repository. You need to download ipsa_sched.c and ipsa_sched.h to be able to run it. The other files are there in case your freeRTOS settings are not identical to ours.

Contents

1 Method **1**
1.1 RTOS Architecture Design 1
1.2 Task Implementation 1
1.3 WCET 2
1.4 Period calculation 2
1.5 Tests et validation 4

2 Results **4**
2.1 Task Execution Analysis 4
2.2 Schedulability and System Utilization 4
2.3 System Stability 5
2.4 Conclusions 5

1. Method

1.1 RTOS Architecture Design

The project utilizes FreeRTOS, a popular real-time operating system for embedded systems, to manage multiple tasks with precise temporal requirements. The design leverages the implementation of both periodic and aperiodic tasks with specific priorities to simulate a realistic multitasking environment.

We've created two files to run the various tasks: an ipsa_schec.c file and an ipsa_sched.h file.

1.2 Task Implementation

1.2.1 Periodic Task 1: Status Display

This task aims to simulate continuous system monitoring by displaying the message "Working" every second. It uses `vTaskDelay` to create a fixed delay of 1000 ms, thereby freeing up the processor for other tasks during this time.

1.2.2 Periodic Task 2: Temperature Conversion

This task converts a fixed temperature of 95 degrees Fahrenheit to Celsius. It executes every 700 ms, displaying the result of the conversion. The formula used is $(\text{Fahrenheit} - 32) \times \frac{5}{9}$.

1.2.3 Periodic Task 3: Multiplication of Large Numbers

It computes the product of two large integers, 123456789 and 987654321, every 1 s. This task illustrates the processing of heavy computations in a real-time environment.

1.2.4 Periodic Task 4: Binary Search

Implements a binary search algorithm on a sorted array of 50 elements, consistently searching for the value 25 every 700 ms. This task demonstrates how to integrate classic search algorithms into an RTOS.

1.2.5 Aperiodic Task (Optional)

This task simulates an operation that requires 100 ms to complete. It is used to demonstrate the management of tasks that do not follow a strict periodic interval.

1.3 WCET

The Worst Case Execution Time (WCET) for each task has been estimated based on preliminary measurements. These estimates are used to define task periods and configure the scheduler to ensure that all tasks can be executed within their respective time intervals without causing time overruns.

To calculate the WCET, we ran each task a thousand times and noted the longest execution time (i.e. the WCET).

1.3.1 Periodic Task 1: Status Display

The selected WCET for the task 1 is **0.0965s**

1.3.2 Periodic Task 2: Temperature Conversion

The selected WCET for the task 2 is **0.1040s**

1.3.3 Periodic Task 3: Multiplication of Large Numbers

The selected WCET for the task 3 is **0.1085s**

1.3.4 Periodic Task 4: Binary Search

The selected WCET for the task 4 is **0.1175s**

1.3.5 Aperiodic Task (Optional)

The selected WCET for the aperiodic task is **0.2984s**

1.4 Period calculation

To calculate the specific values for the task periods, we need to follow an approach based on their WCET and priorities, while ensuring that the sum of the task utilizations does not exceed 1 (or 100%). This is based on the schedulability test for fixed-priority systems.

Task utilization can be calculated as follows: $\sum_{i=1}^{n} \frac{WCET}{Period}$ where n is the number of periodic tasks. However, as a simple starting point, we'd like the total sum of uses not to exceed 0.69 for $n = 4$, to allow time for the aperiodic task to complete.

We know that

- WCET of aperiodic task = 0.30s (highest priority, but managed differently)
- WCET of periodic tasks: Task 1 = 0.10s, Task 2 = 0.11s, Task 3 = 0.11s, Task 4 = 0.12s

We have rounded all WCETs to the nearest hundredth of a second to ensure that we never exceed the WCET.

We're going to assign periods to periodic tasks so as to maximize utilization while remaining schedulable.

Step 1: define priorities

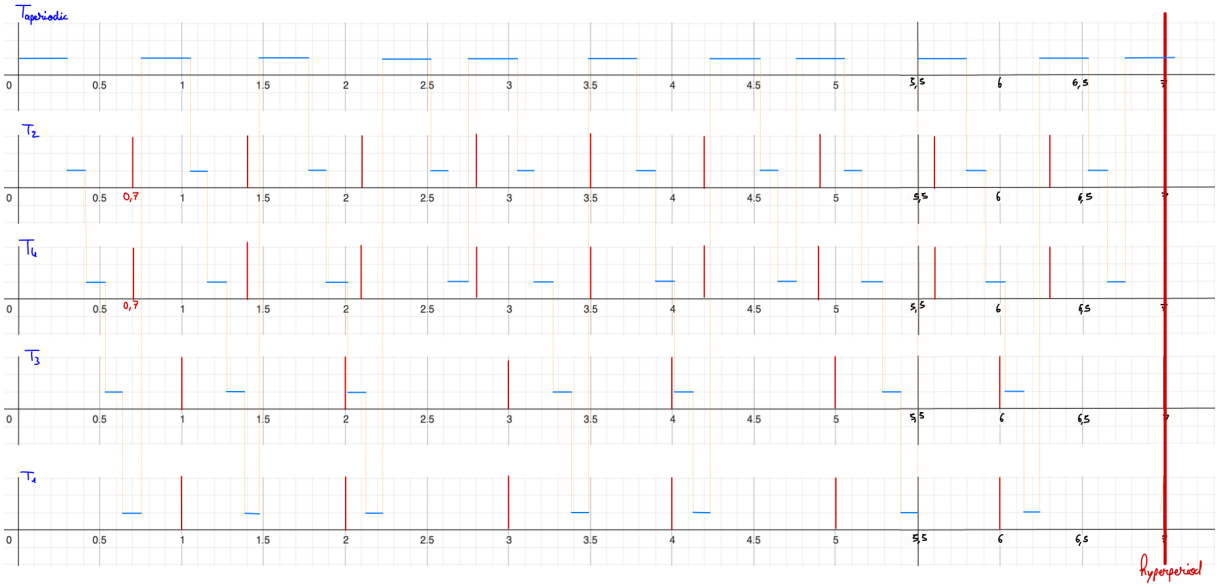
We want the aperiodic task to be performed on every cycle, as it could represent something necessary that occurs via a trigger external to the system and must be processed.

We also want temperature measurement (task two, which converts Fahrenheit to degrees) to be the system's second priority. Its execution frequency must be less than 1s.

Task 4, corresponding to binary sorting, is next. Its execution frequency must also be less than 1s.

And finally, task 1, which displays "working", must be executed no more than once per second.

Step 2: Period calculations Let's now proceed to calculate the periods for the periodic tasks based on this information. After many trials, we obtain the following scheduler :



After reducing the periods as much as possible for tasks 2, 3 and 4, we obtain the following periods

- Task 1: Period = 1 s
- Task 2: Period = 0.7 s
- Task 3: Period = 1 s
- Task 4: Period = 0.7 s

This gives an adjusted total utilization of : $\frac{0.10}{1.00} + \frac{0.11}{0.70} + \frac{0.11}{1} + \frac{0.12}{0.70} = 0.54$. So we have time for the execution of the aperiodic task without the risk of exceeding the system's capacity.

1.5 Tests et validation

Unit and integration tests were carried out to validate each feature. Tasks were run in a simulated environment to observe interactions and correct potential anomalies before deployment on target hardware.

Furthermore, FreeRTOS is a system that, by default, is set to "preemptive" when we try to switch it to "non-preemptive", so an error message appears. We therefore chose to leave the system preemptive, even though our code design does not require a preemptive system.

2. Results

Upon executing the tasks within the designed RTOS using FreeRTOS, the expected behavior was observed with all tasks completing their designated operations within their calculated periods. The trace logs, which capture the execution sequence and completion of each task, demonstrate the system's ability to handle periodic and aperiodic tasks concurrently while adhering to the specified scheduling requirements.

2.1 Task Execution Analysis

The aperiodic task, given the highest priority, executed consistently with a completion message every 100 milliseconds, as designed. The successful frequent execution of this task without causing system overloads validates the aperiodic task's period and priority assignment. This indicates the system's responsiveness to tasks simulating external events requiring immediate attention.

The periodic tasks exhibited the following behavior:

Task 1 (Status Display): Printed "Working" once every second, confirming the expected 1s period.

Task 2 (Temperature Conversion): Accurately converted the temperature from Fahrenheit to Celsius every 0.7 seconds, matching the assigned period and demonstrating the system's timely processing capability.

Task 3 (Multiplication of Large Numbers): Completed the computation and printed the result, indicating the appropriateness of the 1 seconds period for computationally intensive tasks.

Task 4 (Binary Search): Successfully performed the binary search every 0.7 seconds, ensuring that search algorithms are executed within the time constraints.

2.2 Schedulability and System Utilization

The task utilization was calculated and adjusted to ensure a total system utilization of 0.54, well within the acceptable limit to maintain system stability and responsiveness. The trace logs provided a visual confirmation that none of the tasks exceeded their worst-case execution times (WCET), thereby verifying the theoretical schedulability analysis with practical results.

2.3 System Stability

The system's stability was consistently maintained throughout the execution phase. No overruns were detected, and the aperiodic task did not impede the execution of periodic tasks. The proper functioning of all tasks without precedence or deadline violations further underscores the successful implementation of the scheduler.

2.4 Conclusions

The observed task behavior aligns with the theoretical analysis, demonstrating that the tasks were well within the system's schedulable capacity. The periods assigned to the tasks facilitated a balance between task responsiveness and system utilization, resulting in a stable and efficient RTOS environment for the given set of tasks.