The Realtime operate system in the Anti-lock Braking System

Author: Yuzhe Ma

a1918429@adelaide.edu.au

Supervisor: Marian Mihailescu

marian.mihailescu@adelaide.edu.au

**Content**

[**ABSTRACT** 2](#_Toc194265916)

[**KEYWORDS** 2](#_Toc194265917)

[**1 INTRODUCTION** 2](#_Toc194265918)

[**2 Problem Analysis** 2](#_Toc194265919)

[**2.1 Ongoing tasks cannot be released** 2](#_Toc194265920)

[**2.2 Methods and improvement** 2](#_Toc194265921)

[**2.2.1 Priority Inheritance** 3](#_Toc194265922)

[**2.2.2** **Priority Ceiling Protocol** 3](#_Toc194265923)

[**3 Project Plan** 3](#_Toc194265924)

[**3.1 Build experiment** 3](#_Toc194265925)

[**3.1.1 Low-priority** **task** 3](#_Toc194265926)

[**3.1.2 Medium-priority** **task** 3](#_Toc194265927)

[**3.1.3 High-priority** **task** 3](#_Toc194265928)

[**3.2 Using Priority Inheritance and Priority Ceiling Protocol to fix this problem** 3](#_Toc194265929)

[**3.2.1 Priority Inheritance** 3](#_Toc194265930)

[**3.2.2 Priority Ceiling Protocol** 4](#_Toc194265931)

[**REFERENCE** 4](#_Toc194265932)

**ABSTRACT**

This project will analyse the problem of some RTOS in ABS. During the experiment stage, it will simulate the working environment of the vehicle's central processing unit (CPU). With the two methods that have been developed, this project aims to address the problem.

**KEYWORDS**

Realtime operate system(RTOS), Anti-lock Braking System(ABS)

**1 INTRODUCTION**

The anti-lock Braking System (ABS) is one of the most essential systems in a modern vehicle. ABS can reduce stopping distance and prevent wheel lockup when people suddenly apply sharp braking (Oppenheimer, 1988). With ABS, the security of driving is higher. Using RTOS to build ABS makes it more reliable because RTOS is efficient in application management, availability, and its fail-free nature (Pavlo et al., 2024). In RTOS, tasks are typically scheduled based on priority, with high-priority tasks expected to execute before lower-priority tasks. However, the priority reverse problem has a significant impact on ABS. The priority reverse problem causes the low-priority task to be completed before the high-priority task. This issue will become a significant problem in the ABS due to strict timing constraints.

**2 Problem Analysis**

The problem analysis graph will analyse an existing classical problem in a special example and attempt to provide some methods for solving these issues.

**2.1 Ongoing tasks cannot be released**

This problem is often caused by medium-priority tasks. In this problem, there are typically three types of tasks: low-priority, medium-priority, and high-priority tasks. Low-priority tasks share the same resources and data as high-priority tasks (Babaoğlu et al., 1993). For example, a low-priority task is recording speed and mileage data into a travel log. This task shares the same data with the high-priority tasks, which are ABS and emergency brake orders. Then, a medium-priority task refreshes the dashboard. Before the vehicle is started, it records speed and mileage data into a travel log. When people are driving, the medium-priority task will occupy CPU resources and processes. The low-priority task cannot be released because the medium-priority task deletes the stop command. This leads to low-priority tasks continually consuming data from sensors. At this moment, the high-priority task begins, but it is hindered by low-priority tasks occupying the data. In this case, theongoing tasks cannot be released, and the high-priority task cannot work.

**2.2 Methods and improvement**

There are three main methods to solve this issue in that example.

**2.2.1 Priority Inheritance**

Priority Inheritance ensures that all tasks using the same data get the highest priority, which is the priority of those tasks. This means that the lower-priority task will receive the same priority as the higher-priority task. In that example, the low-priority task will get high priority. The priority of low-priority tasks will be higher than that of medium-priority tasks so that the CPU will execute the release of low-priority tasks first.

**2.2.2** **Priority Ceiling Protocol**

The Priority Ceiling Protocol assigns different data with varying priorities, and this priority is inherited by the task that uses that data. It means that speed data will take priority in that example. When the low-priority task uses the data, it not only obtains the data but also acquires the priority associated with that data. This task will be labelled as a high-priority task when it keeps the data. After the CPU shuts down the task, its priority will revert to low-priority.

**3 Project Plan**

**3.1 Build experiment**

This experiment will be built using threads and mutexes.

**3.1.1 Low-priority** **task**

The low-priority task will commence at the program's beginning and lock the data. It will lock the data during the medium-priority task running until the low-priority task is quitted. It may be shut down by the CPU when a high-priority task starts. If the low-priority task is interrupted, it will resume when the high-priority task is completed. In the simplified model, the function of the low-priority task is to print the value of the data.

**3.1.2 Medium-priority** **task**

The medium-priority task will run after the low-priority task has started. Moreover, it will not stop throughout the entire process. In the simplified model, the function of the medium-priority task is to print “the medium-priority task is running.”

**3.1.3 High-priority** **task**

The high-priority task will run after the medium-priority task has started. In the simplified model, the function of the low-priority task is to increase and print the value of the data.

**3.2 Using Priority Inheritance and Priority Ceiling Protocol to fix this problem**

This experiment will employ two methods to address that issue.

**3.2.1 Priority Inheritance**

Priority inheritance will change the priority of the low-priority task to match that of the high-priority task. It can quit when the medium-priority task has been started.

**3.2.2 Priority Ceiling Protocol**

The priority ceiling protocol will adjust the priority of low-priority tasks to match that of the data. It can quit when the medium-priority task has been started.

**REFERENCE**

Babaoğlu Ö, Marzullo K and Schneider, FB 1993. A formalisation of priority inversion. *Real-Time Systems*, 5(4), pp.285–303. doi:https://doi.org/10.1007/bf01088832.

Openheimer, P 1988. Comparing Stopping Capability of Cars with and without Antilock Braking Systems (ABS). SAE Transactions, [online] 97, pp.313–336. Doi: HTTPs://doi.org/10.2307/44724718.

Pavlo M, Ivan K and Anastasiia K 2024. Choosing the Right RTOS for Your Embedded System | Lemberg Solutions. [online] Available at: https://lembergsolutions.com/blog/choosing-right-rtos-your-embedded-system.