ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 11, Issue 4, pp: (1-4), Month: October - December 2023, Available at: www.researchpublish.com

Real-time Clock Periodic Tasks Concurrent Executor

Kin Seng, Lee

Kuala Lumpur, Malaysia

DOI: https://doi.org/10.5281/zenodo.8402201

Published Date: 03-October-2023

Abstract: **Real-time clock periodic tasks concurrent executor program is being written to execute multiple timed periodic tasks concurrently using modern CPU multithreading and multicores feature at specific RTC periodically. By using RTC as an absolute reference, we can analyze the status of multiple isolated devices on selected time instances to know about overall system status or each individual systems status. Multithreading program able to execute multiple tasks concurrently, making RTC periodic tasks possible. While single threaded sequential program only run multiple tasks in sequence, one at a time. Each task runtime creates delays resulting subsequent periodic tasks no longer synchronous. Nevertheless, this program can run without RTC. It becomes an isolated system, capable of performing its own synchronous operations.**

Keywords: **multithreaded application, multithreading, real-time clock, concurrency, concurrent execution, thread pool, periodic tasks, RTC periodic tasks, RTC ticking, critical system, sampling rate, synchronous operations, synchronous, data analysis.**

I. INTRODUCTION

Periodic tasks are tasks which are executed at specified time intervals, again and again, with minimal human intervention; RTC periodic tasks are tasks which are executed repeatedly & automatically if timed at specified real-time clock intervals.

Periodic tasks can be tasks of reading various types of sensory data, on / off relays, or executing software operations such as transmits data.

Modern electronic systems generally have multiple hardware digital & analogue input & output; modern computers are capable of executing multiple software programs currently. When a system's multiple hardware sensors or multiple isolated systems are being sampled at the same time instance; systems hardware relays were triggered at the same time instance; software operations being executed at same time instance, these sensory data & operation outcomes often inherited strong correlations. Even if they are not executed at the precise real-time clock. We may able to conclude overall status of a system or identify any abnormalities among multiple isolated systems at that time instance. In contrast, when multiple data are sampled at different time instance; software or hardware operations are executed at different time instance. These data & operation outcomes typically have weak correlations and we may not able to derive substantial conclusion about the systems.

Concurrent executor capable of executes multiple timed periodic tasks concurrently, hence making it useful in synchronous systems.

Use periodic tasks to periodically reset Watchdog Timer (WDT) to detect and recover from computer malfunction. When Watchdog Timer elapsed, the timeout signal is used to perform automatic software or hardware correction of temporary software / hardware faults.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 11, Issue 4, pp: (1-4), Month: October - December 2023, Available at: www.researchpublish.com

II. MATERIALS AND METHODS

Download the source code from my web URL homepage, compile it using Microsoft Visual Studio and execute to reproduce the reported outcome. The source codes are written in Microsoft C#.

After compilations, execute the application. Microsoft operating system 'Command Prompt' window appear. We can see that task 'A' is executed every RTC second; task 'B' is executed every 2 seconds; task 'C', 'D', 'E' are being executed every 3, 4, 5 seconds respectively.

The homepage for My online source code URL are:

(i) https://www.algoonline.net/Continuously_On-Off_device/Continuously_On-Off_device.htm

In this program, periodic tasks are the functions to be executed by the computer when timed.

In standard software procedure, whenever software threads are being created, they should be ended using 'Thread.Join()' after finished execution. However, Periodic tasks are tasks which are executed at specified time intervals, again and again. Thus, the software worker threads are being placed inside while loop, they will not be ended until we manually set 'RunProgram' variable to false. Moreover, these software threads are not bind to specific tasks. Every second whenever there are any timed period tasks, executor program will assign equivalent count of software threads to execute them.

'Thread.Wait()' are used to allow software worker threads to idle when no tasks to execute. Hence, we can re-use the software worker threads without need to repeatedly create & close them. 'Thread.Wait()' is one of the software thread states, other states are 'Thread.Start()', 'Thread.Run()', and etc.

In this example program, every second software object 'Simulated_RTC_tick' will execute software object 'SortedList_PeriodicTasks.notify_timed()'. The object will check if latest periodic tasks are timed. Assuming software object 'Simulated_RTC_tick' is an accurate RTC that tick every 1 second.

Periodic tasks' function is saved into 'class_record_PeriodicTask' array. To get to know which periodic tasks are timed, we use ascending order 'SortedList' class. 'SortedList' class represents a collection of key/value pairs that are sorted by the keys and are accessible by key and by index. Keys are all periodic tasks next targeted RTC to be executed; values are the 'class_record_PeriodicTask' array index.

Just a reminder. If declared periodic tasks' function as 'SortedList' value, abnormal execution would occur.

Whenever periodic tasks are timed, 'SortedList_PeriodicTasks' software object can wake-up software worker threads from idle to execute timed periodic tasks. Software worker threads are reused without need of create new software threads & exit after finished execution.

We don't have to declare a dedicated worker threads for every periodic task, that would be too overwhelming. Declare sufficient count of software worker threads to ensure that all the timed periodic tasks can be executed concurrently. Vice versa, if the count of periodic tasks is small, we can declare a dedicated worker threads for every periodic task.

Worker threads only execute tasks' function. Users should write the function to save the result data separately.

Moreover, a separate 'tasks' Watchdog Timer timeout' check function will check periodically if any periodic tasks having long execution time due to errors, set those frozen tasks' 'isTimeout' parameter variable to TRUE. Timeout check function checks every task's current execution time periodically, deriving from current time minus tasks' start execution time. User may write task dedicated 'corrective action stage 1' function to mitigate these timeout tasks.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 11, Issue 4, pp: (1-4), Month: October - December 2023, Available at: www.researchpublish.com

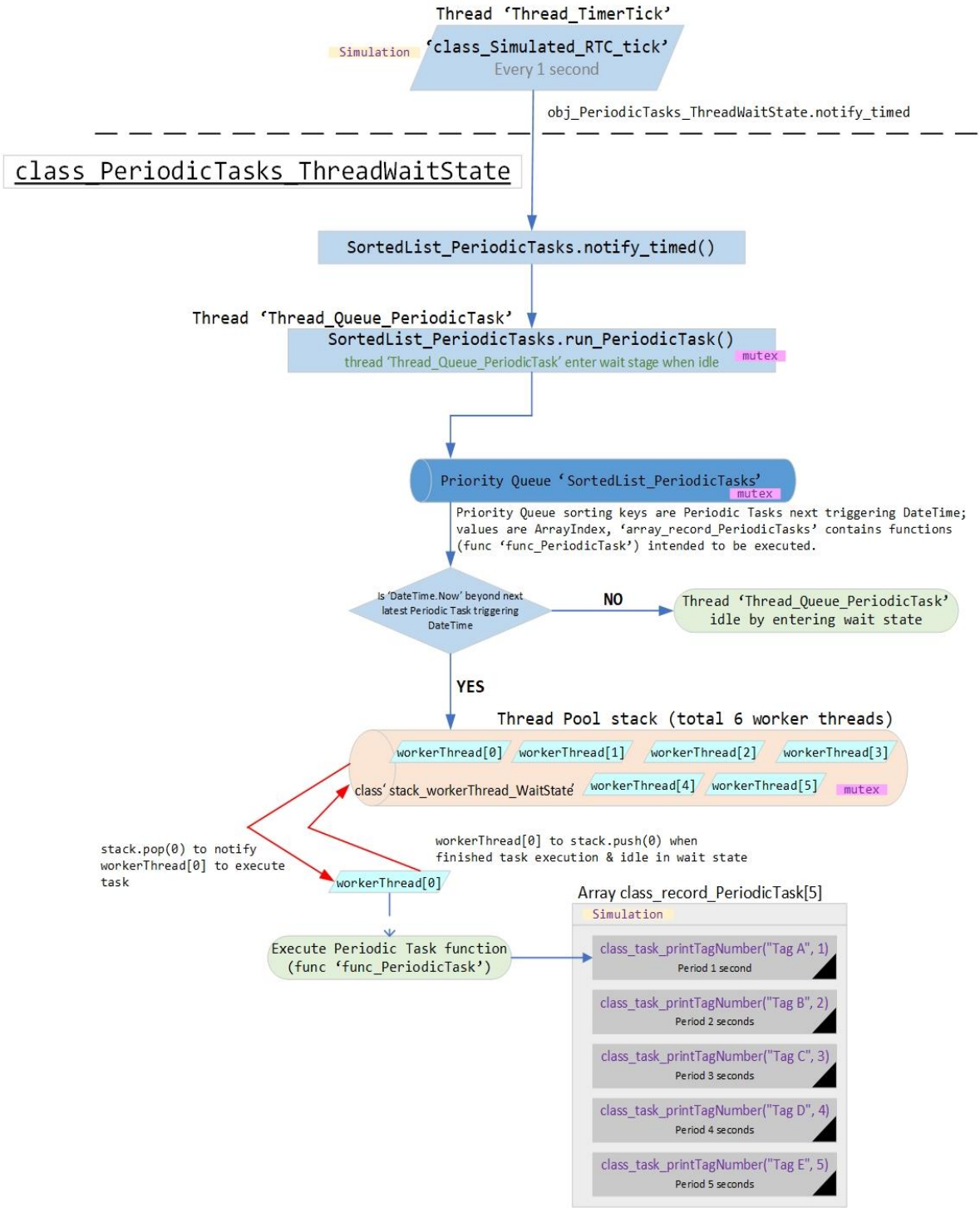

Fig. 1: Process flow diagram

III. CONCLUSION

Electronic systems are getting more & more complicated, they are having more & more hardware inputs & outputs, sensors & relays. Computers are having more & more software installed. Inevitably we would want these periodic tasks to be execute concurrently.

Modern multithreaded CPU and this program empower systems to perform more accurate synchronous actions. Human walking is an example of synchronous actions. We hear & see our surrounding environment & react during our walk; we walk by synchronously moving our legs & swinging our limbs.

In addition, when we make systems' clock adhere to Real-Time Clock (RTC), we can plausibly make multiple isolated systems to perform synchronous operation on the selected time instances. Data analysis become meaningful when multiple isolated systems' data are sampled at the same RTC.

Software watchdog timer to facilitate automatic correction of temporary hardware or software faults and avoid disruption of operations.

REFERENCES

[1] Mark J. Price (2022): C# 11 and .NET 7 – Modern Cross-Platform Development Fundamentals: Start building websites and services with ASP.NET Core 7, Blazor, and EF Core 7, 7th Edition – Packt Publishing 199p, 249p.

[2] Deitel, H.M., Deitel, P.J., Nieto, T.R. (2002): Visual Basic. NET: how to program. – Prentice Hall 720p.

[3] Shakti Tanwar (2019): Hands-On Parallel Programming with C# 8 and .NET Core 3: Build solid enterprise software using task parallelism and multithreading. – Packt Publishing 35p.

[4] Nicolas Halbwachs (1992): Synchronous Programming of Reactive Systems (The Springer International Series in Engineering and Computer Science, 215) 1993rd Edition – Springer 140p.

[5] Björn Lisper (1989): Synthesizing Synchronous Systems by Static Scheduling in Space-Time (Lecture Notes in Computer Science, 362) 1989th Edition. – Springer 111p.