Automated Scheduling Technique Based on Multiple Timer Interrupts for Time-Triggered Co-operative Architecture

S. Kuankid and A. Aurasopon

***Abstract*—Time-triggered system provides more attractive options for many safety-related and safety-critical embedded systems. The work is mainly concerned with developing novel scheduling algorithms and implementation techniques which can be automated and ensured predictability during the process of time-triggered co-operative architecture. The major objective of this work is to modify an automated scheduling technique for use with time-triggered co-operative based on the employment of multiple timer interrupts. The results show that proposed algorithm provides the effective schedulability and can help in a significant reduction of scheduling time as compared with a traditional scheduler.**

***Index Terms*—Time-triggered architecture, time-triggered co-operative scheduler, multiple timer interrupts.**

I. Introduction

This paper is specifically concerned with software development for resource-constrained systems. When designing for small embedded device for safety-related and safety critical systems, it is generally recognized that the use of "Time-Triggered" (TT) system architectures offers significant advantages over other system approaches [1], [2]. In TT architecture, all tasks are activated at a specific time instants based on a periodic timer [3]. Such architecture is statically computed before the system begins to execute. As a result, the TT system will offer a highly-predictable behavior and can suit for many safety-related applications, such as anti-lock brakes, airbags, or biomedical sensors [4], [5].

In TT designs, there has been interested design option which simple, generate a complete and highly-predictable schedule called Time-Triggered Co-operative (TTC) scheduler. Such scheduler also known as cyclic executive scheduler-describes as a useful design pattern that can be used to build in resource-constrained embedded system. The TTC can be applied in various automotive applications, a wireless communication system, various control application systems, data acquisition systems, washing-machine control, and monitoring of liquid flow rates [1], [2].

Despite many advantages, TTC might be suffered from failure modes that can greatly impair system performances which are the fragility of scheduler, the problems of task jitter, and task overruns [5]. To avoid suffering from inappropriate task scheduler, there has been alternative approached by developing the scheduler implementation based on the technique of "Multiple Timer Interrupt" (MTI) called "TTC-MTI" scheduler, this technique can be addressed such problems effectively. In literature, Kuankid *et al.* [3] indicated that TTC-MTI scheduler achieved better performance in timing behavior as opposed to the other technique. However, in practical terms, there has been a few studies explore the automated scheduling technique to process of such scheduler.

According to the aim of this paper, the work has interested to modify an automated time-triggered scheduling technique based on MTI scheduler for use with a range of time-triggered co-operative architecture, to ensure that user can select appropriate task scheduler when implementing with the technique based on multiple timer interrupts. This paper is organized as follows: Section II gives related work of scheduling in real-time system. Section III presents the automated scheduling technique of TTC scheduler based on MTI technique. Section IV finds the method and results from experiment. Finally, Section V concludes the use of automated scheduling technique.

II. Related Work

The main components while developers create software for use in real-time system is the task scheduler. When categorizing scheduling algorithm based on the triggering mechanism, there are two common approaches used in scheduling real-time systems: Event-Triggered (ET) system and time-triggered system [4, 5].

In the event-trigger system, all tasks are invoked as a response to events when external events take place [5]. In this case, the system is controlled purely by the response to external events, typically represented by interrupts which can arrive at any time [4]. Event-triggered system is generally recommended for application which offers high responsiveness, flexibility, and ability to handle sporadic data are exchanged in the system [4]. However, due to unknown request time from the external events, such system might suffer in less predictable and poor determinism. As for time-triggered system, all tasks is activated at specific time instants under the control of a timer [4, 6, 7]. The system is usually driven by a global clock which generates periodic interrupts from hardware timer that overflows at specific time instants. The system can suit for many control applications which offers a highly-predictable behavior. In addition, such system is preferred a choice of safety-related system in which all system activities must be known during the design phase [4], [5], [7]. According to this work, there has interested software development for safety-related system. Therefore,

Manuscript received June 8, 2016; revised October 12, 2016.

The authors are with the Department of Electronics Engineering, Faculty of Science and Tecnology, Nakhon Pathom Rajabhat University, Thailand (e-mail: apinan.a@msu.ac.th).

DOI: 10.7763/IJET.2017.V9.989

the automated task scheduling is designed based on the time-triggered architecture.

When developing software with TT architecture, there has been interested design option called TTC scheduler. In TTC architecture, all tasks in the system is run to completion, one task cannot pre-empt other tasks and only one timer interrupt is supported. Fig.1 shows the operation of TTC scheduler. In these circumstances, Task A executes while Task B has to wait until Task A to completion, and returns to control the scheduler. After that, Task B will be executed. As can be seen that the TTC can be avoided conflicts over the problems of share resources and task can communicate safety by means of global variables [7].

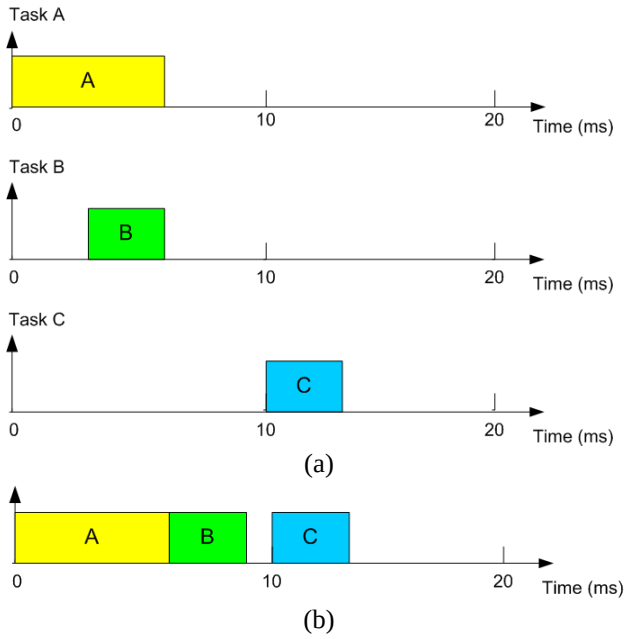

Fig. 1. (a) A schematic representation of three tasks which need to be scheduled. (b) The operation of a typical TTC scheduler

In literatures, the TTC has a wide range of possible implementation options available, these different options have varying resource requirements and performance behavior. The simplest approach for implementing TTC scheduler in low-cost embedded devices is TTC-SL scheduler (Time-Triggered Co-operative - Super Loop) [8]. This architecture is implemented based on the super loop, can be seen simple, easy to implement, and very small resource requirements. However, the TTC-SL approach is not sufficient reliability for precise time. It also operates at full-power because of inefficient use of idle mode. An alternative solution to this problem is TTC-ISR architecture (Time-Triggered Co-operative - Interrupt Service Routine) [7]. The TTC-ISR scheduler can be created using "Interrupt Service Routine" linked to the overflow of a hardware timer. The timer is set to overflow at regular "tick interval". When tick interval occurs, the system will execute task following the task scheduler. The main advantage is that the successive function calls will take place at precisely-defined intervals. However, because the system is not separate between the "scheduler" code and the "application" code (i.e. tasks), if this scheduler runs multiple tasks, particularly tasks with different periods, the system will become not simple and may be difficult to debug and/or maintain. Another problem is that if the task exceeding it predict execution time, the system may be ignored the timer tick interval, this can generate a domino effect on the subsequent tasks, and causes the system hang indefinitely. The alternative solution is TTC-Dispatch scheduler (Time-Triggered Co-operative-Dispatch), this scheduler can be created using ISR the same as TTC-ISR. To avoid losing ticks from task overruns, this TTC-Dispatch scheduler separates the timer ISR and the process of task execution. In TTC-Dispatch scheduler, the software employs two principal functions, which are Update and Dispatch function. The Update function uses to update the scheduler at regular time intervals and Dispatch function uses to organize tasks to be executed when they are due to run. The operation of TTC-Dispatch scheduler is illustrated in Fig.2 When the interrupts occur, the ISR will be called the Update function. In this Update function, the scheduler sets appropriate flags in order to note that an interrupt has occurred. After Update function has completed, a Dispatch function will be called, and the identified tasks will be executed following in sequence. The Dispatch function is called from the Main function similar to TTC-ISR scheduler. The system is usually placed in a Sleep function (idle mode) in order to reduce system operating power after complete execution each task.

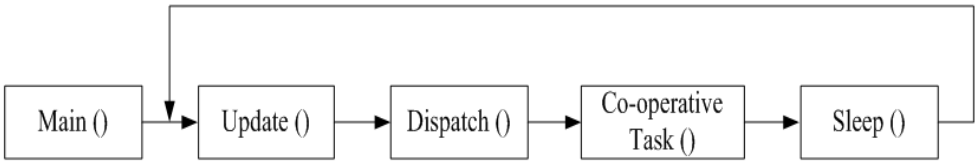

Fig. 2. Function call tree for the TTC-Dispatch scheduler (adapted from [2])

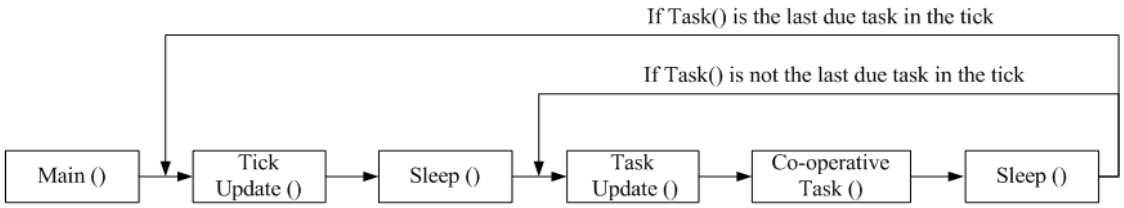

Fig. 3. Function call tree for the TTC-MTI scheduler (adapted from [9]).

III. Scheduler Implementation Based on the Technique of Multiple Timer Interrupts

More recently, there has been alternative approached to fixed problems of TTC scheduler by developing the scheduler implementation based on the technique of multiple timer interrupts called "TTC-MTI" scheduler [2], as shown in Fig. 3, the process using two interrupts update functions which are Tick Update (ISR tick interrupt) and Task Update (ISR task interrupt) function. The Tick Update function is used to generate the periodic tick interval whereas Task Update is used to notify for execution task within the period of tick interval.

In general, the Tick Update function which is called every tick interrupt will arrange the task which ready to execute within the current periodic tick interval. If the system has task to execute in this period, the scheduler will set up and enable timer following the required release time of task interrupt, after that placing the processor to the idle mode (Sleep function). However, if within this period has no task to execute, the process will disable task interrupt and go to the Sleep mode in order to reduce power consumption within this period.

When task interrupt occurring, the Task Update function will run the first task until completion, after that checks the other tasks in this tick interval. If this period has other tasks to run, the process will set up timer for the next task interrupt and execute all tasks until completion before placing the processor to the sleep mode. After all tasks within the tick interval were executed successfully, the scheduler will disable the task interrupt, then place the processor to Sleep function, and waiting for executing with the next tick interval.

IV. Modified Automated Scheduling Technique of TTC Scheduler

In TTC architecture, the tasks are organized to execute following the types of implementation that are TTC-SL, TTC-ISR, TTC-Dispatch, and TTC-MTI. It can be seen that effective scheduling techniques are the need for the TTC architecture because an inappropriate task scheduling may cause all tasks set cannot be scheduled at all. Moreover, if the scheduler is implemented improper behavior, this problem can lead to high levels of task release jitter and also increased power consumption in the system [9]-[11].

As for TTC-MTI scheduler, Kuankid *et al*. [3] proposed the scheduling algorithm which implemented based on multiple timer interrupts to test the schedulability of system. Such algorithm will compute the system that feasible to schedule or not. This can help in a significant reduction of design stage before the system execution. Therefore, to fulfill the gap between the scheduling algorithm and scheduler implementation, this paper proposes to modify an automated time-triggered scheduling technique for use with TTC-MTI scheduler. This system can help in applying the scheduler in real-time system, especially in cases that user wants to add or remove tasks to the system while operating.

In the implementation phases, the automated task scheduling can be modified as show in Fig.4. In general, the scheduler will execute task and return to run the scheduler as the pattern of the previous section. On the other hand, if user wants to add or remove tasks, the processor will disable both of the tick interrupt and task interrupt to stop the scheduler. Then calculate the schedulability of all tasks following the scheduling algorithm [3]. In this computation, if all tasks can be scheduled, the process will set up the new scheduler and add or remove task to the system, after that enable the tick interrupt to start a new scheduler. However, if all tasks is not feasible, the process will return to run the previous scheduler again.

V. Experimental Methodology and Results

To evaluate the performance of automated task scheduling, it presents scheduling time and schedulability test of proposed system by comparing with a traditional scheduler.

A. Scheduling Time

1) Hardware platform and software development tools

In this experiment, the target platform is a small microcontroller LPC2129. The LPC2129 is based on a 32 bit ARM7 microcontroller, which is used an oscillator frequency of 12 MHz and a CPU frequency of 60 MHz. The CPU consists of two 32 bit timers with 4 multiple channels in each timer. Accordingly, this CPU has enough timers to implement the TTC-MTI scheduler. As for the software development, this paper used development tools from Keil products[12]. The tool chain was used RealView MDK version 4.12.

2) Task specifications

To explore the performance of this algorithm, task set parameters which consists of 100 tasks were randomly generated with standard uniform distribution and the random task set can be scheduled at all. By assuming all tasks are period, the deadline is equal to its period. The worst case execution time and period of all tasks is generated according to the following inequalities:

$$0 < WCET(i) < 100?\ us \tag{1}$$

$$WCET(i) < P(i) < 10000\ us \tag{2}$$

3) Results

The experiments were tested and compared the performance between traditional algorithm and proposed TTSA-MTI algorithm. The result in Table I shows that the scheduling time of TTSA-MTI is significant reduction in computation time when compare with a traditional approach.

B. Schedulability Test

1) Task specifications

For meaningful testing, task set parameters which consists of 10 tasks were randomly generated with standard uniform distribution at different utilization (0-1) following the inequalities 1 and 2.

2) Results

The results of percentage of schedulable can be shown in

Fig. 5. It can be seen that the performance of proposed algorithm is closely related to the traditional approach. However, the traditional approach is slightly higher the proposed algorithm at the utilization more than 0.75

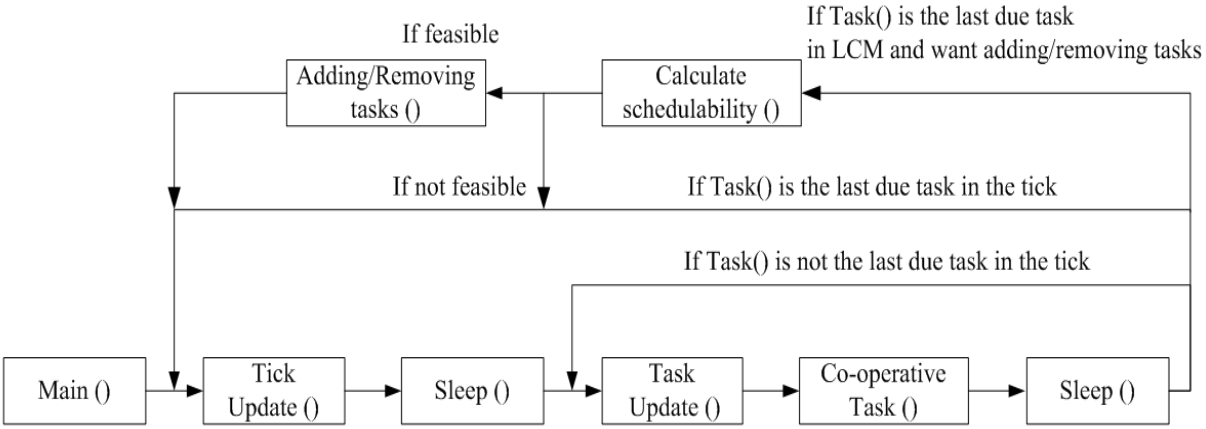

Fig. 4. Function call tree of automated task scheduling of TTC-MTI scheduler

C. Scheduling Time

1) Hardware platform and software development tools

In this experiment, the target platform is a small microcontroller LPC2129. The LPC2129 is based on a 32 bit ARM7 microcontroller, which is used an oscillator frequency of 12 MHz and a CPU frequency of 60 MHz. The CPU consists of two 32 bit timers with 4 multiple channels in each timer. Accordingly, this CPU has enough timers to implement the TTC-MTI scheduler. As for the software development, this paper used development tools from Keil products[12]. The tool chain was used RealView MDK version 4.12.

2) Task specifications

To explore the performance of this algorithm, task set parameters which consists of 100 tasks were randomly generated with standard uniform distribution and the random task set can be scheduled at all. By assuming all tasks are period, the deadline is equal to its period. The worst case execution time and period of all tasks is generated according to the following inequalities:

3) Results

The experiments were tested and compared the performance between traditional algorithm and proposed TTSA-MTI algorithm. The result in Table I shows that the scheduling time of TTSA-MTI is significant reduction in computation time when compare with a traditional approach.

D. Schedulability test

1) Task Specifications

For meaningful testing, task set parameters which consists of 10 tasks were randomly generated with standard uniform distribution at different utilization (0-1) following the inequalities 1 and 2.

2) Results

The results of percentage of schedulable can be shown in Fig.5. It can be seen that the performance of proposed algorithm is closely related to the traditional approach. However, the traditional approach is slightly higher the proposed algorithm at the utilization more than 0.75

TABLE I: THE SCHEDULING TIME BETWEEN TTC-DISPATCH AND TTSA-MTI SCHEDULER

Tasks	TTC-Dispatch (ms)			TTSA-MTI (ms)		
	Mean ± SD	Maximum	Minimum	Mean ± SD	Maximum	Minimum
10	2.83 ± 0.014	2.85	2.81	0.144 ± 0.002	0.146	0.141
20	9.77 ± 0.015	9.80	9.75	0.317 ± 0.001	0.319	0.315
30	20.77 ± 0.015	20.80	20.76	0.513 ± 0.002	0.515	0.510
40	35.77 ± 0.017	35.80	35.75	0.733 ± 0.001	0.735	0.730
50	54.87 ± 0.015	54.89	54.85	0.979 ± 0.002	0.983	0.977
60	77.95 ± 0.018	77.98	77.92	1.248 ± 0.002	1.251	1.247
70	105.21 ± 0.014	105.23	105.19	1.547 ± 0.001	1.549	1.545
80	136.37 ± 0.011	136.39	136.35	1.865 ± 0.002	1.868	1.863
90	155.20 ± 0.018	155.23	155.17	2.352 ± 0.002	2.357	2.350
100	187.80 ± 0.020	187.83	187.78	2.622 ± 0.001	2.623	2.620

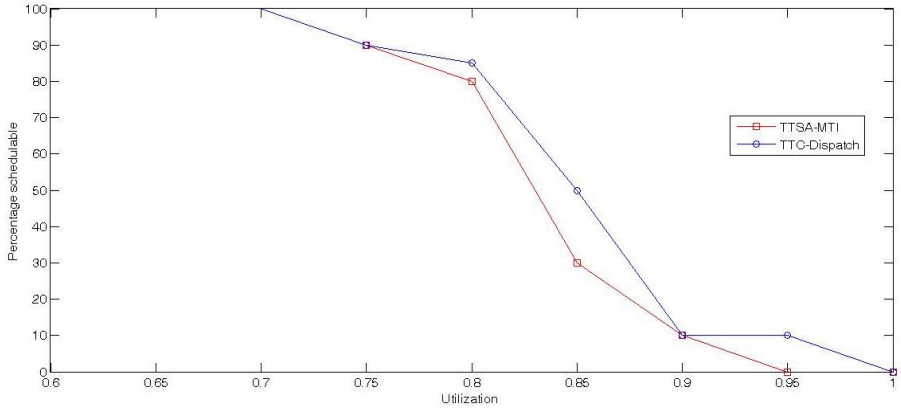

Fig. 5. Schedulability test between TTC-dispatch and TTSA-MTI scheduler at different utilization.

VI. Conclusions

Due to the predictability and highly reliable behavior of TTC architecture, this work concentrates to implement such scheduler in real-time resource-constrained embedded system. The work is mainly concerned with developing novel scheduling algorithms and implementation techniques which can be automated and ensured predictability during the process of TTC architecture. The results show that proposed algorithm provides the effective schedulability and can help in a significant reduction of scheduling time as compared with a traditional scheduler.

Acknowledgment

This research was financially supported by Mahasarakham University (2015) Copyright of Mahasarakham University.

References

[1] A. K. Gendy and M. J. Pont, "Automatically configuring time-triggered schedulers for use with resource-constrained, single-processor embedded systems," *IEEE Trans. Electron Devices*, vol. ED-11, pp. 34-39, Jan. 1959.

[2] M. Nahas, "Employing two 'sandwich delay' mechanisms to enhance predictability of embedded systems which use time-triggered co-operative architectures," *Journal of Software Engineering and Applications*, vol. 04, pp. 417-425, 2011a.

[3] S. Kuankid *et al.*, "Effective scheduling algorithm and scheduler implementation for use with time-triggered co-operative architecture," *Elektronika ir Elektrotechnika*, vol. 20, pp. 122-127, 2014.

[4] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications: Kluwer Academic*, 1997.

[5] A. Albert, "Comparison of event-triggered and time-triggered concepts with regard to distributed control systems," in *Proc. Embedded World*, Nurnberg, Germany, 2004.

[6] J. W. S. Liu, *Real-Time System*, Prentice Hall, 2000.

[7] M. J. Pont, *Patterns For Time-triggered Embedded Systems: Building Reliable Applications with the 8051 Family of Microcontrollers*, ACM Press Books, 2001.

[8] S. Kurian and M. J. Pont, "Restructuring a pattern language which supports time-triggered co-operative software architectures in resource-constrained embedded systems," presented at the EuroPLoP 2006, 2006.

[9] M. Nahas, "Bridging the gap between scheduling algorithms and scheduler implementations in time-triggered embedded systems," Doctor of Philosophy, Department of Engineering, University of Leicester, Leicester, 2008.

[10] A. Maaita, "Techniques for enhancing the temporal predictability of real-time embedded systems employing a time-triggered software architecture," Doctor of Philosophy, Department of Engineering, University of Leicester, Leicester, 2008.

[11] T. Phatrapornnant and M. J. Pont, "Reducing jitter in embedded systems employing a time-triggered software architecture and dynamic voltage scaling," IEEE, 2006.

[12] *Keil Embedded Development Tools*, 2012.

S. Kuankid was born in Thailand. He received a B.Eng. degree in electronics engineering from King Mongkut Institute of Technology Ladkrabang in 1999, a M. Eng. degree in electrical engineering from King Mongkut's University of Technology Thonburi in 2004, and a Ph.D. degree in electrical and computer engineering at Mahasarakham University in 2015, Thailand. He is currently a lecturer in department of electronics engineering, Faculty of Science and Tecnology, Nakhon Pathom Rajabhat University, Thailand. His main topic of research is closely related with a background of embedded microprocessor system, information and communication technology, and renewable energy applied in agricultural systems.

A. Aurasopon was born in Amnat Charoen Province, Thailand, in 1971. He received his B.Eng. degree in electronic engineering form the Northeastern College, Khon Kaen, Thailand, in 1995, and his M.Eng. and Ph.D. degrees in electrical engineering form the King Mongkut's University of Technology Thonburi, Bangkok, Thailand, in 2003 and 2007, respectively. He was a lecturer in the Department of Electrical Engineering, Faculty of Engineering, Burapha University, Chonburi, Thailand, in 2007. He transferred to the Faculty of Engineering, Mahasarakham University, Maha Sarakham Province, Thailand, in 2008, where he is presently an associate professor. His current research interests include soft-switched converters, ac choppers, converter systems for improving harmonics, power factor and the application of electronics and computers to agriculture.