International Journal of Computer Science and Informatics

Volume 1 | Issue 2 Article 1

October 2011

Analysis Of Time Triggered Schedulers In Embedded System

E. Anbarasi
M.E Pervasive computing Technologies, Anna University of technology, Tiruchirappalli, India,
anbuek@gmail.com

N. karthik
nkarthikapce@gmail.com

Analysis Of Time Triggered Schedulers In Embedded System

E.Anbarasi[1] N.karthik[1]

[1]M.E Pervasive computing Technologies, Anna University of technology, Tiruchirappalli, India.
[1]Email:{ anbuek ,nkarthikapce}@gmail.com

Abstract:

In embedded software, scheduler is one of the essential component. It is essential to schedule the task in a suitable way to achieve the system optimization. Time triggered scheduling approaches render a reliable performance for real time systems. This paper sketches on time triggered schedulers which tends to results in highly predictable system behavior. Despite many scheduling approaches, time triggered cooperative and time triggered hybrid schedulers have more attention in resource constrained embedded system which works on single processor. Our target of this work is to analyze these time triggered schedulers' performances. We evaluated and studied the performance of those time triggered schedulers with various parameters. The results have been incurred by implementation of schedulers in embedded c programming language with simulation tool.

Keywords*: schedulers; time triggered co-operative; time triggered hybrid.*

I. INTRODUCTION:

Embedded system can be viewed as a processor to perform one or more dedicated function with computing constraints. Most of the embedded systems are real time system for producing the output within the predefined time limit. The embedded system works in resource constrained environment. So the proper scheduling mechanism helps the system to function reasonably. Scheduler can be specified as a set of coding that allows the process or task directed to the CPU for execution. Scheduler can be defined as a simple operating system that allows tasks to be called periodically or on a one shot basis [6]. The allocation of the task to the processor depends on the type of scheduling methodology.

The two common approaches used in scheduling embedded systems are event triggered and time triggered. In event triggered, tasks are involved as a response to events represented by the external interrupts. On the other hand, the primary substitute to the event triggered system is time triggered system where the tasks are involved periodically under the control of the timer.

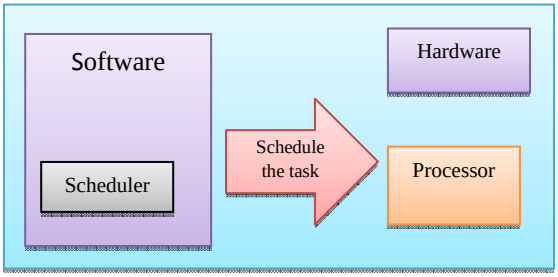

Fig.1 Embedded System

Here we are considering the time triggered cooperative (TTC) and time triggered hybrid (TTH) scheduling coming under the time triggered approach often applied in the low cost control system and safety related application where the task characteristics are known priori. The primary reason for preferring the time triggered approach for the application is that the results in the system are highly predictable in nature.

In this paper we have analyzed the performance of time triggered co-operative and time triggered hybrid schedulers by using the following parameters: Design of scheduler, power saving mode, memory allocation, and response to external events, run time overhead, choice of execution, scalability, portability and flexibility.

Organization of the paper:

The remainder of this paper can be organized as follows. Section II clarifies about the various time triggered scheduling approaches and scheduler components. The parameters and system environment are reported in the section III. Section IV explains about the analysis and result. Finally

Section V having some concluding remarks and future work.

II. TIME TRIGGERED SCHEDULING APPROACHES:

We will concern here with the two main types of time triggered schedulers such as time triggered co-operative and time triggered hybrid.

Time triggered co-operative (TTC) schedulers:

In TTC Scheduler each ready task cooperates to let the running one task to finish. None of the task does a block anywhere from the start to finish. The set of co-operative task is executed at a predefined time intervals from the scheduler tick. However the next start of scheduling is among the ready task that runs in turn only from a priority wise ordered list. The scheduler is implemented using a hardware timer, which is used to generate interrupts on a periodic basis. A co-operative Scheduler provides single task system architecture. Tasks are scheduled to run at specific time either on a periodic or one shot basis. Only one task is active at any point of time. There is no response from the co-operative scheduler if any external event occurs. TTC scheduler provides low jitter features [3]. Power saving mechanisms like Dynamic Voltage Scaling is used with TTC and same low level jitter is maintained.

TTC scheduler is driven by periodic interrupts generated by the on chip timer. When an interrupt occurs, the processor executes an update function. In the update function, the scheduler checks out to see if any tasks are due to run and set appropriate flags. After the checks are complete, a dispatch function will be called and the identified tasks are executed. The system will enter into a low power (idle) mode.

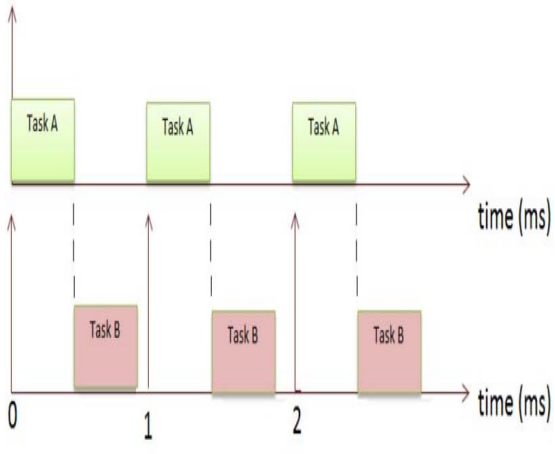

Fig.2: TTC scheduler

Time triggered hybrid (TTH) schedulers:

A single, time triggered preempting task can be added to TTC Architecture to give what we have called a time triggered hybrid (TTH) scheduler [1]. A Hybrid Scheduler provides limited multitasking capabilities. It supports many numbers of co-operative tasks and it can be preempted by a single task. The interrupt can be used to preempt the co-operative tasks. The hybrid scheduler gives rapid response to the external events. The implementation of the hybrid scheduler is simple and contains small amount of code. It allocates the memory for two tasks at a time. The preemptive task cannot itself be interrupted by any of the co-operative tasks. The preemptive task has higher priority than co-operative tasks. If the preemptive task interrupts and find that a critical section, it will use the resources by using lock, after completion of the execution it will release the lock. The co-operative task will then resume and find the system in the same state.

Use of TTH architecture allows us to create a static schedule made up of a collection of tasks which operate co-operatively and a - pre-emptive task. The pre-emptive task will be used for periodic data acquisition, typically through an analogue-to-digital converter or similar device: such a requirement is common in, for example, a wide range of control systems.

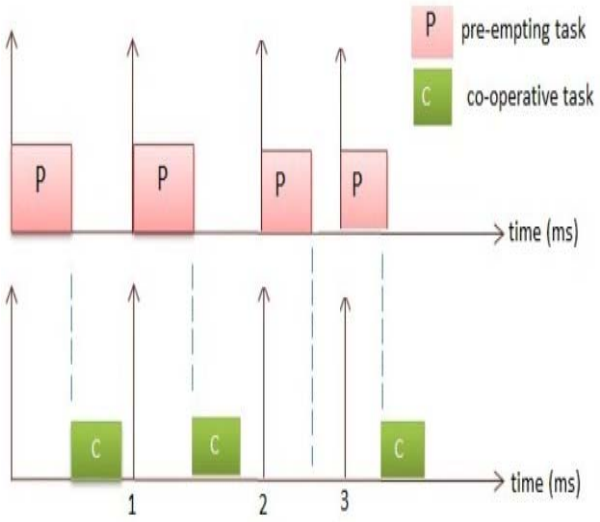

Fig.3: TTH Scheduler

Scheduler components:

The time triggered scheduler (TTC and TTH) components are shown below.

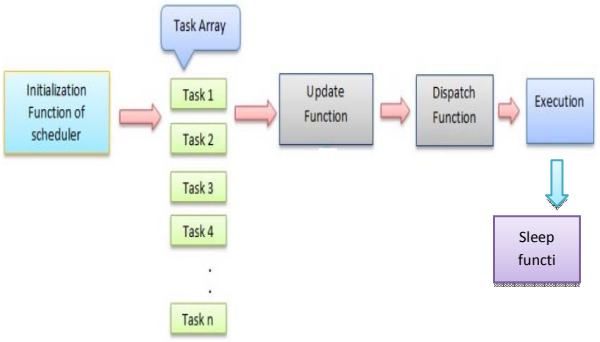

Fig.4 Components and Function of Time Triggered Scheduler

Heart of the scheduler is the scheduler data structure. The data structure of the scheduler is a user defined data type which collects the information about all tasks in the scheduler. The task array consists of list of tasks which are needed to execute by the scheduler.

The scheduler requires an initialization function to prepare the scheduler task array and error code operation. The main operation of this function is to set the timer to generate the regular ticks that will drive the scheduler. This function is invoked by the overflow of the timer. It is the scheduler interrupt service routine. If the task is due to run, then it increments the RunMe flag. Then it will be executed by the dispatcher function. Add Task Array is used to add the tasks to the array.

The add task function carries three parameters. They are task name, initial delay of the task and the period of the task. It returns the position of the task in the task array. The dispatcher function is used to send off the task for execution. After all tasks are added in the task array, this function is called to begin the scheduling process.

The Start Function is very simple and its starts the scheduler by enabling the interrupts. Delete Task Function is to delete the task from the task array after its execution. Go to Sleep Function is used to reduce the power consumption of the system. Sleep function put the system to sleep or idle mode at the end of the dispatcher call, it will then wake up when the next timer tick occurs. An 8 bit error code variable is used to report the errors at any part of the scheduler. It is called from update function.

III. PARAMETERS AND SIMULATION ENVIRONMENT:

The following parameters are employed to examine the performance of the time triggered schedulers approach for the embedded systems.

1. **Design of scheduler**:
   Developing a scheduler with reliable operating environment that matches precisely the needs of most embedded applications. It includes the selection of the programming language for the design of the scheduler.
2. **Power saving mode**:
   An important feature of scheduled applications is that they contribute themselves to low power operation. This is achieved through one of the power saving mode called “idle “mode where the CPU activity is halted but the state of the processor is sustained. In this mode, the power required to do operation with processor is 50% reduced than the normal mode.
3. **Memory allocation**:

This parameters explains about the space needed for the code size of the scheduler and the tasks in the task array. It also includes the allocation of memory by the processor to schedule the task at a run time.

4. **Response to external events:**

This determines with the handling of pre-emptive task by the scheduler. Interrupts are used to preempt the task in the schedulers. It may be a hardware or software interrupt. Usually the hardware timer is used to generate the interrupt; if it overflows the interrupt will be generated to preempt the task in the scheduler. This parameter explains how fast the scheduler can responsive to the external task.

5. **Runtime overhead:**

It refers to the scheduling methodology and synchronization of the task by scheduler at execution time. Run time overhead occurs due to the context switching of the task and retrieval of partially computed results.

6. **Choice of execution**:

Usage of schedulers according to the application of the system. This parameter involves the selection of the platform in which the scheduler can be implemented. Depends on the application, the choice of execution will vary.

7. **Portability and flexibility**:

Portability is refers to reuse the existing code instead of creating new code when moving scheduler from an environment to another. That is a scheduler created for one microcontroller family member may be ported for use on any other type of microcontroller device.

8. **Scalability:**

It tells about number of tasks that can be supported by the scheduler with predictable response. Even if the number of tasks gets increased, the performance of the scheduler should not degrade.

Simulation Environment:

We have used Keil development tool µversion V4.00a which supports 8051 microcontroller family, for simulation of the codes. The Embedded C language is used for development of embedded software.

IV. ANALYSIS:

Using performance analyzer(PA) in the keil µversion V4.00a, we can check out the execution time and average execution time for processes. First we consider the TTC method for scheduling which are co-operative in nature. In TTC we took three processes for scheduling. Then we found the execution time of the processes by setting the count to 10. Similarly we found the execution and average execution time of the TTH Scheduling by considering three processes with single preemptive task.

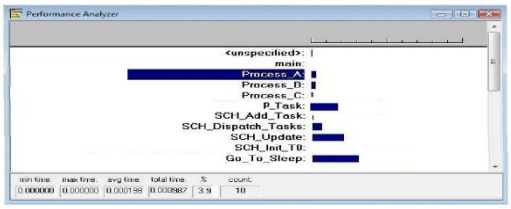

Fig.5: PA analysis of TTH Scheduler

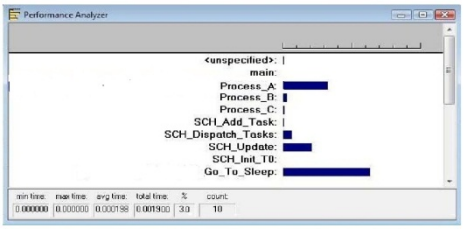

Fig.6: PA analysis of TTC Scheduler

Results:

The time triggered co-operative scheduler implementation is simple and it can be carried out with embedded c programming language. The applications which use the TTC scheduler are more reliable and safe. The TTH Scheduler has the ability to deal with single short frequent task and long infrequent tasks. From the above analysis of TTC and TTH scheduler, the parameters which determine the performance of the schedulers are listed below.

V. CONCLUSION:

The first approach Time triggered co-operative scheduler is executed and it can be implemented in low cost control applications. But it has the drawback of that the current execution task cannot be halted by other task. The TTH scheduler is implemented in resource constrained safety related applications for checking of errors in emergency conditions. This approach can ensure that the application is able to respond to the external event within the particular time limit. In most of the time triggered schedulers, an offline (static) schedule is said to be best choice. In future, we are going to work on assigning the priority for the preemptive task dynamically at the run time in time triggered hybrid scheduler.

Parameters/ Approaches	**Time Triggered Cooperative scheduler**	**Time Triggered Hybrid scheduler**
Design of scheduler	Simple, Reliable	Reliable if carefully designed
Power Saving Mode	Supports Idle mode	Supports Idle mode
Memory allocation	One task at a time	Two task at a time
Response to external events	No response	Rapid response
Runtime overhead	Low	High
Choice of execution	Low cost control applications	Resource constrained applications
Portability/ Flexibility	more	minimum flexibility
Scalability	Required number of tasks can be added	More number of tasks can be added

References:

[1]. Michael J Pont, "Patterns for Time Triggered Embedded Systems", TTE-Systems, Pearson-Education, April 2008.

[2]. Michael J Pont, Embedded C, Pearson Education, 2002.

[3]. Susan Kurian and Michael J. Pont, Huiyan Wang and Teera Phatrapornnant," Selecting an appropriate scheduler for use with time-triggered embedded systems "Embedded Systems Laboratory, University of Leicester.

[4]. Ayman K. G. Gendy, University of Leicester," Techniques for scheduling time-triggered resource-constrained embedded systems", June 2009.

[5]. B.Madhusudan Rao, M. KeerthiTeja, N.Nitin, " Comparison of Process Scheduling Methodologies for Embedded Systems ", ICETET-09.

[6]. Pont, M.J. (2001) "Patterns for Time Triggered Embedded Systems: Building Reliable Applications with the 8051 Family of Microcontrollers", Addison Wesley/ACM Press.

[7]. Ayman K. Gendy and Michael J. Pont,"Automatically Configuring Time-Triggered Schedulers for Use With Resource-Constrained, Single-Processor Embedded Systems", IEEE transactions on industrial informatics, vol. 4, no. 1, february 2008

[8].Michael L. Pinedo, Scheduling Theory, Algorithms and Systems,Third Edition, Springer, Printice Hall, 2008.

[9].Arnold S. Berger, Embedded Systems Design :An Introduction to Processes, Tools and Techniques, CMP Books, 2008.

[10]. Qing Li and Carolyn Yao , Real Time concepts for Embedded Systems, CMP Books, 2003. Peter Brucker, Scheduling Algorithms, Fifth Edition, Springer, 2007.