Performance Analysis of Open-Source Real Time Operating Systems

J.Ramprabu, G.Sindhuja

Abstract – This paper presents the importance of analysis and evaluation of Performance Metrics of Real Time Operating systems. Advanced parameters such as Semaphore Shuffling Time, Deadlock Breaking Time, Task-Task message Passing Time are considered in this paper. A 32-bit Microcontroller is chosen for benchmarking. Two RTOS that are available as open source for educational purpose are considered.

Index Terms – open source RTOS, performance evaluation, performance measurement, RTOS analysis

I. INTRODUCTION

Embedded Systems is widely flourished technology of the world today. They play a vital and inevitable role in Aerospace engineering, Automotive electronics, Entertainment Industry, Medical Electronics, etc. These domains can be divided as non time-critical systems and time-critical systems. Time-critical systems focus to meet the deadline on time without compromise, taking into account the Best case and the Worst case scenario of execution. The main factors that contribute in achieving deadlines are the speed of the processor in executing instructions and the Real Time Operating System running on the microcontroller. The RTOS is a platform that can run the time-critical application and contributes to the scheduling of applications and task management. Real Time Operating System is one of the critical aspects of any embedded system that has time-critical application running on it.

This paper is deals with the following sections. The RTOS chosen for evaluation and its features are explained in second section. The importance and definition of benchmarking parameters are discussed in the third section. The Methodology used and the evaluation results are considered in the fourth and fifth section respectively. The conclusion is described in the final section.

J.Ramprabu *Assistant Professor, Department of Electrical and Electronics Engineering, Kumaraguru College of Technology, Coimbatore -641049.*

G.Sindhuja , *Second Year, M.E. (Embedded Systems), Department of Electrical and Electronics Engineering, Kumaraguru College of Technology, Coimbatore -641049.*

II. RTOS CHOSEN FOR EVALUATION

RTOS is generally implemented in any Embedded System for an application to run to ensure that the application's execution time become predictable and deterministic.

In this paper, FreeRTOS and µCOS-II are the two RTOS considered for evaluation in this paper. These RTOS are open-source and free for educational purpose. The kernels are based on the Priority based Pre-emptive Scheduling algorithm. So the evaluation is based on apples-to-apples comparison. Both of these RTOS are commonly used in Safety-critical applications.

A. FreeRTOS

FreeRTOS is developed by Real Time Engineers Ltd. It provides both Pre-emptive and Co-operative Scheduling [2], [3]. FreeRTOS does a deterministic operation and has a memory footprint of 10K at the maximum. FreeRTOS currently extends support to 35 microcontroller architectures. The number of APIs is less and functionalities are more.

B. MicroController/OS-II

MicroController/OS-II (widely known as µC/OS-II) is a real-time kernel used in Avionics, Medical Equipments, Consumer Electronics, Automotive Electronics etc. The kernel is developed by Jean J. Labrosse. It provides Preemptive Scheduling and multitasking real-time kernel. The memory footprint can range from 6KB to 24KB. µC/OS-II can support upto 254 application tasks [1], [4].

III. BENCHMARKING PARAMETERS

Benchmarking techniques are developed to facilitate engineers with techniques to evaluate and compare the RTOSs running on any embedded systems. The definition of the parameters considered for evaluation is explained below.

A. Semaphore Passing Time

Asynchronous events such as External Interrupts are synchronized with the RTOS Scheduler using Semaphores. Semaphores can be Binary or Counting Semaphores. Hence Semaphore Passing Time is the delay between the time when a task releases its semaphore and the time when another task acquiring this Semaphore getting to running state. This

ISSN: 2278 – 1323

parameter demands atleast four tasks in the ready state. The tasks must be of different priorities from each other.

Semaphores are also used to allocate resource to a particular task when multiple tasks demand for same resource. The semaphore passing time quantifies the overhead related to mutual exclusion. When multiple tasks compete for the same resources this is expected.

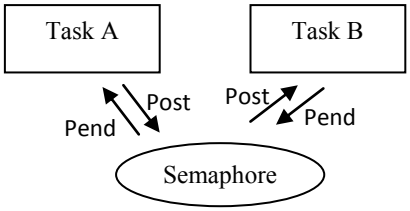

Fig 1. Semaphore Passing Time

B. Deadlock Violation Time

Deadlock Violation is expected during a higher-priority task preempting a lower-priority task, especially when low-priority task holds a resource needed by the higher-priority task. Deadlock Violation Time measures the average time taken to resolve this violation.

It is normal to anticipate deadlocks in preemptive multitasking kernels. Dynamic assignment of priority is the effective way to resolve this problem. The running task priority is raised above that of the interrupting task till the resource to be shared is released by the lower priority task. Later the priority can be lowered. Deadlock violation time, hence, is calculated as the sum of times required to achieve resource sharing between a low priority task acquiring a resource and a higher-priority task needing it.

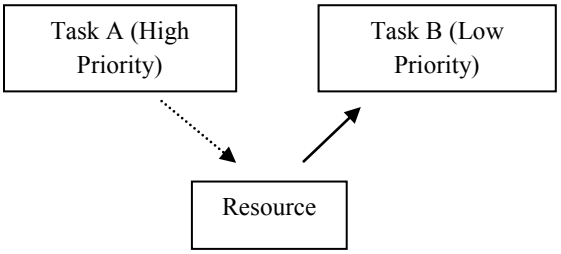

Fig 2. Deadlock Violation Time

C. Task-Task message Passing Time

Task-Task Message Passing Time is the delay within the application when a definite length message is passed from one task to another. The following is the algorithm to calculate this delay. There has to be atleast two tasks to share a message. In this case, the sending task should go to blocked state immediately after sending the message and the receiving task should be suspended until receiving this message. The Task-Task message-passing connection is to be established dynamically. When multiple data are sent on the same connection, the receiving task can read an old message before the sending task overwrites it with a new one.

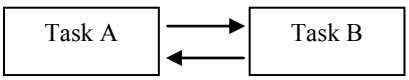

Fig 3. Task-Task message Passing Time

IV. METHODOLOGY

The benchmarking of the above said parameters are done based on Rhealstone Benchmarking techniques. The above said parameters are advanced when compared to the basic inevitable parameters such as Task Jitter, Scheduler Jitter, Task Latency and Interrupt Latency, Context-Switching Times. The Scheduler tick frequency, Preemptive Mode of Scheduling, Interrupt priority and Task Priorities are maintained the same throughout the evaluation.

V. EVALUATION RESULTS

The evaluation of the advanced parameters results in the average values taken. The average is calculated based on repeated execution of the test for fifteen iterations.

TABLE 1
Evaluation Results of open-source RTOS

	FreeRTOS	µC/OS-II
Semaphore Passing Time	0.333µs	0.207µs
Deadlock Violation Time	40.14µs	41.31µs
Task-Task message Passing Time	2.25µs	3.06µs

IV. CONCLUSION

Thus the evaluation of FreeRTOS and µC/OS-II on the 32-bit Microcontroller Platform based on the chosen measurement metrics is performed. The results are presented above. These results conclude that both the RTOS maintain the consistency. Hence they prove to be suitable for any time-critical applications.

REFERENCES

[1] J. J. Labrosse, "µC-OS/II the real-time kernel", R & D Books, (Miller Freeman, Inc.), Lawrence KS, 1999.

[2] R. Barry, "A portable, opensource mini real-time kernel", http://www.freertos.org, Oct. 2007.

[3] http://www.freertos.org/RTOS.html

[4] http://micrium.com/rtos/ucosii/overview/

ISSN: 2278 – 1323

[5] R. Kar. and K. Porter. "Rhealstone: A Real-Time Benchmarking Proposal". Dr. Dobb's Journal, 1989.

[6] D. Kalinsky, "Basic concepts of real-time operating systems", LinuxDevices magazine, Nov. 2003.

[7] J. Ganssle, "The challenges of real-time programming", Embedded System Programming magazine, vol. 11, pp. 20–26, Jul. 1997.

[8] K. Baynes, C. Collins, E. Filterman, B. Ganesh, P. Kohout, C. Smit, T. Zhang, B. Jacob, "The performance and energy consumption of embedded real-time operating systems", IEEE Transactions on Computers, vol. 52, pp. 1454–1469, 2003.

[9] K. Curtis, "Doing embedded multitasking with small microcontrollers, part 2", Embedded System Design Magazine, Dec. 2006.

[10] A. Martinez, J.F.Conde, A.Vina, "A comprehensive approach in performance evaluation for modern real-time operating systems", Proceedings of the 22nd EUROMICRO Conference, pp. 61, 1996.

[11] K. M. Sacha, "Measuring the real-time operating system performance", Seventh Euromicro workshop on Real-time systems proceedings, Odense, Denmark, pp. 34–40, Jun. 1995.

[12] G. Hawley, "Selecting a real-time operating system", Embedded System Design Magazine, 1999.

[13] M.Timmerman, L.Perneel, "Understanding RTOS technology and markets", Dedicated Systems RTOS Evaluation project report, Sep. 2005.

[14] Segger, "EmbOS real-time operating system user & reference guide", Segger Microcontroller System GmbH, 2006.

[15] D.Stewart, "Echidna real-time operating system", http://www.glue.umd.edu/dstewart/serts/research/echidna/index.shtml.

[16] A. J. Massa, "Embedded software development with ECOS?" Prentice Hall, Nov. 2002.

[17] G. C. Buttazzo, "Hartik: A hard real-time kernel for programming robot tasks with explicit time constraints and guaranteed execution", Proceedings of the 1993 IEEE International Conference on Robotics and Automation, Atlanta, Georgia, USA, pp. 404–409, May 1993.

[18] Keil, "Keil real-time kernel and operating system", http://www.keil.com/rtos/, Mar. 2007.

[19] R. Chrabieh, "Operating system with priority functions and priority objects", TechOnline technical paper, Feb. 2005.

[20] B. Millard, D. Miller, C. Wu, "Support for ADA intertask communication in a message-based distributed operating system", Computers and Communications Conference Proceedings, pp. 219–225, Mar. 1991.

[21] Renesas Technology Corp., "Renesas high-performance embedded workshop (HEW)",http://www.renesas.com/fmwk.jsp?cnt=ide_hew_tools_product_landing.jsp&fp=/products/tools/ide/ide_hew/, 2007.

[22] Renesas Technology Corp., "M16c/62P group hardware manual", http://documentation.renesas.com/ eng/products/mpumcu/rej09b0185_16c62pthm.pdf, Jan. 2006.

[23] Micrium, "µC-OS/II ports for Renesas microcontrollers,"http://www.micrium.com/renesas/index.html, Nov. 2007.

[24] Segger, "EmbOS trial version for M16C6X (HEW 4.0 with NC30 version 4.00)",http://www.segger.com/downloadform_embos_m16c_nc30_v540.html, Nov. 2007.

[25] I. Ripoll, P. Pisa, L. Abeni, P. Gai, A. Lanusse, S. Saez, "RTOS state of the art analysis", Technical report, Open Components for Embedded Real-time Applications (OCERA) project, 2002.

[26] D. Kalinsky, "Asynchronous direct message passing rapidly gains popularity", Embedded Control Europe Magazine, Nov. 2004.

Prof.J.Ramprabu *works as an Assistant professor in the Department of EEE, Kumaraguru College of Technology. He has an Academic experience of 8 years. He is currently Pursuing research in Anna University Chennai and his area of working is renewable energy and embedded system. He is a Member of the Indian Society for Technical Education and Member of System society of India.*

G.Sindhuja *received B.E in Electronics and Communication Engineering from Sri Shakthi Institute of Engineering and Technology, Coimbatore. She is presently pursuing Post Graduation in Embedded Systems from Kumaraguru College of Technology.*

ISSN: 2278 – 1323