MALAYSIAN SCIENCE & TECHNOLOGY CONGRESS '98
Symposium C: Computer Science & Information Technology
Universiti Sains Malaysia, Pulau Pinang, 10 -11 November 1998

EVALUATION OF REAL-TIME OPERATING SYSTEM FOR SMALL-SCALE EMBEDDED SYSTEMS

Dyg. Norhayati Abg. Jawawi, Safaai Deris

Faculty of Computer Science and Information System,
Universiti Teknologi Malaysia,
Karung Berkunci 791, 80990 Johor Bahru
Email : dayang@mail.fsksm.utm.my

Rosbi Mamat

Faculty of Electrical Engineering,
Universiti Teknologi Malaysia,
Karung Berkunci 791,
80990 Johor Bahru

ABSTRACT

In this paper, the performance of some real-time operating systems for small-scale embedded systems are evaluated based on some criteria. The evaluation is performed qualitatively and quantitatively. The evaluation results based on a case study on an engineering application will be presented.

Keywords : *Real-time system, embedded system, real-time operating system*

1. Introduction

As technology in microelectronics advances, more and more microprocessor-based system or embedded systems are being used in many application areas. Embedded systems are widely used in communication, office equipment, automotive systems and household appliances. Also, embedded systems have becoming more sophisticated and implements many functions in one product, frequently, they must work in real-time. As a result, software development for real-time embedded systems are growing in scale and becoming very complex over the years. The need for fast development turnaround time has made improving software productivity an urgent need.

In a real-time system the correctness of the system depends not only on the logical results, but also on the time at which the results are produced [3]. A real-time system is inherently concurrent and multitasking since it has to react to and process numerous events simultaneously. Real-time systems differ from the traditional data processing systems is that they are constrained by non-functional requirements such as dependability and timing [1]. Thus, developing software for even small-scale embedded real-time systems can be very difficult.

In order to increase the software productivity, maintainability and flexibility in developing real-time systems, special software tools such as a real-time operating system (RTOS), proper software engineering methodologies and high-level programming language need to be considered.

Therefore, the main objective of this paper is to presents some results in evaluating the commercially available RTOS for a typical embedded real-time system, namely a wall climbing robot controller which is been developed at the Faculty of Electrical Engineering, Universiti Teknologi Malaysia [2].

The layout of this paper is as follows. In Section 2 a review of RTOS and its importance in real-time software development is briefly described. Section 3 discussed the wall climbing robot controller for which the real-time software is to be developed. The evaluation results of commercial RTOS which are suitable to be used in the wall climbing robot control software are presented in Section 4. Finally, the conclusion is given in Section 5.

2. Real-time Operating System

Real-time operating system (RTOS) is an integral part of a real-time multitasking system. The four main functional areas that they support are process management and synchronization, memory management, interprocess communication and input-output [4].

Since the RTOS is responsible for intertasks communication and memory management, the use of RTOS relieves the embedded systems developer of the need to worry about tasks management and intertasks communication. In the development of real-time embedded systems, the use of a RTOS will increase the software productivity and improve the performance of the real-time system.

To develop a real-time system which uses RTOS, a suitable RTOS must be selected. In the process of selecting the suitable RTOS, some criteria such as: performance and function must match problem requirements, reliability, cost, development environment, compatibility with existing tools, the size of memory and other resources required by the operating system and support from vendor, should be considered.

3. Wall Climbing Robot Controller

The embedded controller for the wall climbing robot can be represented in block diagram form as shown in Figure 1. The main processor in the controller is based on Intel 80188XL microcontroller with 64K EPROM and 64K RAM. The main function of the digital controller is to move the four legs of the robot with a predefined sequence during climbing operation. Each leg consists of three motors, which correspond to the three joints at each leg. Some control algorithms control each motor. The computation of the control algorithm must be completed within the chosen sampling period. Typical sampling period for motor control is 50 milliseconds.

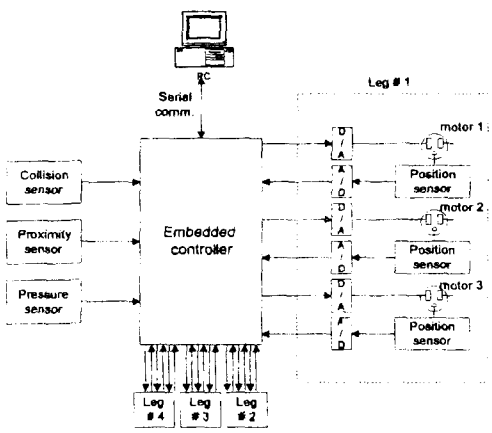

Fig 1 Block diagram of the wall climbing robot controller

The embedded controller monitors its environment using some sensors - collision sensor, proximity sensor and pressure sensor. It also communicates with a remote PC to receive commands and sending back information via a serial communication link.

4. Evaluation of RTOS for a Wall Climbing Robot Controller

Several criteria, specific to the wall climbing robot controller must be considered in the process of selecting a suitable RTOS. Thus, evaluation results presented in this section are based on the criteria specified to the robot controller. However, it was felt that, these results are also relevant to other similar systems.

Since, the main processor used in the robot controller is based on Intel 80188XL microcontroller, it enables the use of good quality and low cost PC based C compilers such as from Borland and Microsoft in the software development. The C language is preferred to be used in the control software development for reasons of productivity and maintainability. As such, the RTOS must be compatible with Intel 80x86 16-bit non-protected mode operation and support PC based C development tools.

The suitable RTOS must also be ROMable with less effort involved in the process, and does not assumed any other resident operating systems such as DOS or WINDOWS. Other criteria includes scheduling policy, fast context switching, enough tasks and system management calls and the cost of the RTOS.

Sixty commercial RTOS compatible with Intel 80x86 processor have been evaluated against the set criteria for the robot controller. It should be noted that, the evaluations were based on information published by the manufacturers [5],[6],[7],[8],[9], as it is very difficult to evaluate the RTOS without having access to them.

Table 1 shows the short listed RTOS which satisfied most of the criteria set for use in the robot control software.

RTOS Name	Source Code Availability	Cost (USD)
Byte-Bos from Byte-Bos Integrated Systems	Yes	$7495
C Executive from JMI Software Systems	No	$2500 - $3750
AMX Real-time Multitasking Kernel from Kadak Products Ltd.	Yes	$1900- $7900
SMX from Micro Digital Inc.	Yes	$3500
RTKernel from On-time Setanker	Yes	Not Available

Table 1 Suitable RTOS for robot controller

Table 2 shows some tasks management information of the short listed RTOS. From Table 1 and Table 2, it can be seen that, SMX RTOS is the cheapest of five and also with the slowest task switching time. Except for C Executive, the other four RTOS supplied the source code either with the product or with an extra cost. The availability of the source code is useful for adapting and modifying the RTOS to suite the target system. This is important in the current work as the software development is done using the PC based C compiler from Borland and ROM locator software.

RTOS Name	Scheduling Policy	Max. Tasks	Task Switching Time
Byte-Bos	Fixed priority, round robin time slice	No limit	100 clock cycles
C Executive	Fixed priority, time slice, dynamic	No limit	50 clock cycles
AMX.	Fixed priority, time slice, dynamic	No limit	Varies
SMX.	Fixed priority, round robin, time slice, dynamic, preemptive	No limit	600 clock cycles
RTKernel	Fixed priority, round robin, time slice	No limit	160 clock cycles

Table 2 Suitable RTOS with information on scheduling policy, maximum tasks supported and task switching time

Out of five RTOS shown in Table 2, SMX is the only RTOS with preemptive scheduling policy. A real-time system is more responsive with a preemptive RTOS. However, the requirement for the responsiveness of the robot control system has not yet been established. Hence, requirement whether to use a preemptive or a non-preemptive RTOS has not been decided.

Technical information from the manufacturers quoted that these RTOS require between 2K to 53K ROM size and between 0.1K to 17K of RAM in order to operate. This requirement is important for a memory critical system such as the robot controller system. Amongst the RTOS discussed here, Byte-Bos requires the least memory to operate.

5. Conclusion

This paper presents some results in evaluating the commercially available RTOS for an embedded wall climbing robot controller which is been developed at the Faculty of Electrical Engineering, Universiti Teknologi Malaysia.

Evaluating RTOS for use in the robot controller was found to be very difficult. There is no clear cut to which one is the best for this particular application. Some RTOS have advantages over the others. Since, the cost is the main factor in this application, it is tempted to select the cheapest RTOS. However, more works are to be carried out in evaluating the RTOS. Currently, works are been done in UTM to evaluate the non-commercial RTOS including the free, shareware and research RTOS.

6. References

1. Burns A., Wellings A. J.: "HART-HOOD: Designing Method for Hard Real-time ADA", Journal of Real-time Systems, Vol. 6, 1994, pp73-114.

2. Mohammed Yassir Mahdi Al-Zaydi: "Locomotion Modeling and Simulation of Quadruped Walking and Wall climbing Robot", Master of Engineering thesis, Faculty of Electrical Engineering, Universiti Teknologi Malaysia, October 1997.

3. Pressman, Roger S.: Software Engineering A Practitioner 's Approach. Fourth Edition. New York, U.S.A.: McGraw-Hill, 1997

4. Ramamritham, K. and Stankovic J. A.: "Scheduling Algorithms and Operating systems Support for Real-Time Systems", inveted paper, Proceeding of IEEE, Jan 1994, pp. 55-67

5. http://www.bytebos.com

6. http://www.jmi.com

7. http://www.kadak.com

8. http://www.on-time.com

9. http://www.smxinfo.com