IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE)
e-ISSN: 2278-1676,p-ISSN: 2320-3331, Volume 12, Issue 3 Ver. V (May – June 2017), PP 01-03
www.iosrjournals.org

A Simple Analytical Organized Information of different RTOS used in Embedded Systems

Nishamary P[1]
[1]*Assistant Professor, ECE, Jyothi Engineering College, Cheruthuruthy, Thrissur, Kerala, India*

Abstract*: The embedded technology has been developed to a higher extent. The rtos has significant role in the development of embedded systems.In this paper different rtos features with respect to the parameters has been enlisted. The quantitative and qualitative analysis about rtos helps the user to select the rtos as per requirements. The real time operating systems implemented in hardware provide significant speed-ups over those implemented in software.*

Keywords*: Rtos, Vxworks, QNX, RTLinux, Rtos parameters*

I. Introduction

An operating system is a set of software that handles computer hardware. RTOS is an operating system dedicated to support real-time operations and is implemented in embedded systems. Embedded systems used today are a complex combination of software and hardware. RTOS are responsible for the overall system performance and task solving methodology. RTOS are multitasking operating systems which not only depends upon the logical correctness but also upon the application delivery time. RTOS have been used in many applications from car, ship and airplane electronics to wireless and optical communication equipment, medical instrumentations, multimedia, internet and even home appliances, factory automation, process control, financial transaction processing and video games machines. There are four main tasks of an operating system: process management, interprocess communication and synchronization, memory management and I/O management. Some of the popular RTOS are quantitatively and qualitatively analysed. [1].RTOS is broadly classified in to three categories hard, soft and firm [1].In hard real time systems the task and interrupt requests must be completed within their required deadlines. In firm real time systems the missing of an infrequent deadlines are tolerable. In soft real time systems is one in which deadlines are mostly met, but this constrains is not very lightly time bound. Selecting any rtos for a specific application is governed by certain parameters that determine its performance.

Virtual and dynamic memory management, Task management, Task synchronization, various platform and usb supportability, Scheduling creation and dispatching of task, suitability for the mission critical approach, multicore /multiprocessor supportability, Interrupt handling Capability, Semaphore Supportability, real time priority levels, and fast track pre-emption are some of the features of RTOS. It is difficult for a designer to select an appropriate RTOS which meet the requirements. The performance parameters for selecting an rtos are Scheduler, Thread Synchronization, Interrupthandler,contextswitching,Intertaskcommunication, Power management, Thread switch latency, No of interrupts, kernelsize, Scheduling Algorithms, Synchronization and Language Support, Maintainance Fee, Timers, Priority, Levels, Kernal Synchronization, Cost, Development host, Task Switching time, Royalty fee. According to embedded market study,the use of an RTOS is required in more than 68% of applications[5].The problem with many software RTOS is that they are independent on the processor's performance and load. Real time operating systems implemented in hard ware provide significant speedups over those implemented in software. Some of the designs for implementing RTOS support in hardware are system in chip RTOS,RTM Unit, HW-RTOS using SMP,ARPA-MT Embedded SMT Processor RTOS Hardware Accelerator and MERASA Project[5].

II. Comparison of Contemporary RTOS Used in Embedded System

A. PSOS

PSOS is a popular real-time operating system that is being primarily used in embedded applications. It is available from Wind River Systems, a large player in the real-time operating system. It is a host-target type of real- time operating system. PSOS consists of 32 priority levels. In the minimal configuration, the foot print of the operating system is only 12KBytes. For sharing critical resources among real-time tasks, it supports priority inheritance and priority ceiling protocols. It support segmented memory management. It allocates tasks to memory regions. A memory region is a physically contiguous block of memory. A memory region is created by the operating system in response to a call from an application[4].

B. VxWorks

VxWorks is a host –target system.The host can be either a Windows or a Unix machine. VxWorks has a multitasking kernel. It uses pre-emptive and round-robin scheduling algorithms. It supports memory protection mechanism which isolates real time process from other user mode application and kernel.It Supports Intel, MIPS, PowerPC, SH-4, and ARM architectures[2][4].

C. QNX

QNX is quite small microkernel operating system. Its kernel contains only CPU scheduling, interprocess communication, interrupt redirection and timers. It Supports PowerPC, x86 family, MIPS, SH-4, and the closely inter-related family of ARM, StrongARM and XScaleCPUs[2].

D. eCOS

eCos stands for Embedded Configurable Operating System. eCos is a non-proprietary customizable real-time operating system. It supports bitmap scheduler and multi priority queue based scheduler. It Supports ARM, CalmRISC, FR-V, Hitachi H8, IA-32, Motorola 68000, Matsushita AM3x, MIPS, NEC V8xx, Nios II, PowerPC, SPARC, and SuperH. It has memory pool based dynamic memory allocation.[2].

E. RT Linux

RT Linux is a free operating system. It is robust and efficient and is also known as real time extension of Linux. It is hard RTOS and has preemptive microkernel that runs the entire linux operating system as a fully pre-emptive process. It has two parts Real time and Non real time.Real time that runs on the RT kernel and Non-real time that runs on Linux. It supports FIFO.It uses regular Linux memory management provisions It Supports the Alpha, ARC, ARM, AVR32, Blackfin, C6x, ETRAX CRIS, M32R, m68k, META, Microblaze, MIPS, MN103, Nios II, OpenRISC, SPARC, x86[2][4].

F. Windows CE

Windows CE is designed for devices that have minimal memory; one megabyte of memory is enough for a Windows CE kernel to run. Devices are often configured without disk storage, and sometimes may be configured as a "closed" system that prevents end-user extension.A feature of Windows CE which makes it distinct from other Microsoft operating systems is that large parts of it is available in source code form. It supports x86, MIPS and 32-bit ARM platforms. It is especially designed for time sensitive embedded systems. It support priority based time slice algorithm. It has large memory mapped file support.

G. Free RTOS

FreeRTOS is a free to embed open source real time operating system which supports about 35microcontroller architectures. Free RTOS supports tick-less mode for low power applications. It supports multithreading using priority based round robin scheduling. It has a minimal ROM footprint. It supportsAR,AtmelAVR,AVR32,HCS12,Microblaze,Cortus, MSP430,PIC,Renesash8/s,SuperH. It provides primitive memory management techniques like allocate and free algorithms with memory coalescence.[2].

H. µC/OS-II

µC/OS-II is a free RTOS, easily available on Internet. It is written in ANSI C and contains small portion of assembly code. µC/OS-II has a fully preemptive kernel. µC/OS-II allows up to 64 tasks to be created. µC/OS-II uses a partitioned memorymanagement [4].

I. Lynx

Lynx is a self-host system. The currently available version of Lynx is a microkernel-based real-time operating system. The Lynx microkernel is 28KBytes in size and provides the essential services in scheduling, interrupt dispatch, and synchronization [4].

III. Characteristics of RTOS Support Hardware

When designed a hardware support for RTOS, It has been seen that HW-RTOS block executes common RTOS system calls in hardware including task scheduling, prioritization as well as managing semaphores and mailbox operations. Bench mark shows that context switching can execute up to 2-3x faster than typical SW-RTOS operation at the same CPU clock speed with less jitter [3]. In the HW-RTOS, there is no tick interrupt. HW-RTOS block in hardware take care of timing management. For HW-RTOS,a running task can be switched: When its internal clock, the OS reference timer, causes pre-emption for a timeout previously called for by a task. The HW-RTOS provides semaphores, mailboxes, flags and mutexesand has OS management calls to put a task to sleep, rotate task precedence, disable OS dispatching [3]. The HW-RTOS has a hard interrupt

mechanism where preregistered service calls can be automatically run when a particular interrupt occurs. These automatic interrupt service calls can be semaphore or flag signalling or to wake up a task. No software is involved. Tasks, semaphores, flags, mutexes, mailboxes etc can be created statically at compile time or dynamically as appropriate at runtime. A HW-RTOS call is made from a task, at which time the kernel sees that another task has a higher priority.

When a task timeout expires and HW-RTOS determines it is time to reschedule, a dedicated interrupt is reserved in the ARM core. The CPU services this interrupt and relays execution to the task selected by HW-RTOS. To increase speed, a user can instead of using software ISR preconfigure the HW-ISR table to perform certain services; signal a flag set, post to a semaphore, or wake up a task. HW-RTOS does not protect against deadlock / priority inversion for mutexes. It release task from waiting, wakeup a task, cancel wakeup, and put the calling task to sleep with timeout option. Comparing with a typical software RTOS operation that is basically sequential, the HW-RTOS is closely tied to the CPU and allows for interrupt handling while not interrupting the current task, and that the performance is not dependent on the number of task switching. By simply doing the system calls through familiar RTOS environments such as uItron or uC/OS-III HWOS, one can easily manage multiple tasks while having the hardware IP do the heavy load of resource management and prioritization[3].

IV. Conclusions

In this paper salient features and usability of various real time operating systems has highlighted. There is a very rich field involving the choice of suitable RTOS for critical and non-critical tasks.HW-RTOS help to improve rtos operation and would be more advantageous for industrial applications. Using the HW-RTOS,the tasks could execute upto 3x faster than typical SW-RTOS operation at same CPU clock speed and less jitter.

References

[1] Jane W. S. Liu, "Real-time System", published by Person Education, first edition.
[2] Mr.SagaJape, Mr.MihirKulkarni, Prof.DiptiPawade*"Comparison of Contemporary Real Time Operating Systems"* International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, Issue 11, November 2015.
[3] Carl Stenquist "HW-RTOS Improved RTOS Performance by Implementation in silicon" Renesas Electronics America Inc,May 2014.
[4] http://www.nptel.ac.in/courses/Webcourse-contents/IIT%20Kharagpur//Embedded%20systems/Pdf/Lesson-32.pdf
[5] http://www.site.uottawa.ca/~mbolic/ceg4131/RTOS%20report.pdf.