VxWorks RTOS

Introduction:

VxWorks is a real-time operating system developed as proprietary software by Wind River Systems. It is a high performance, Unix like, scalable RTOS, and supports ARM, Pentium, Intel X-Scale, Super H and other popular processors for embedded system design. VxWorks design is hierarchical and well suited for hard real time applications. It supports kernel mode execution of tasks for fast execution of application codes. VxWorks is supported with powerful development tools that make it easy and efficient to use. Many simulation tools, time-performance analysis tools, debug and test tools are provided, making VxWorks as an RTOS that supports development of almost any embedded application, providing a complete solution for all stages of the design cycle. The latest version VxWorks 6.9 is the first commercial-grade RTOS to fully support both 32-bit and 64-bit processing on Intel Architecture.

Basic Features:

Some important features of VxWorks RTOS are,

1. Multitasking environment using standard POSIX scheduler
2. Ability to run two concurrent Operating systems on a single processing layer
3. Multiple file systems and systems that enable advanced multimedia functionality
4. Synchronization using a full range of IPC options
5. Different context saving mechanisms for the tasks and ISRs
6. Virtual IO devices including pipes and sockets

7. Virtual memory management functions

8. Power management functions to enhance the ability to control power consumption

9. Interconnecting functions that support large number of protocols.

10. Pipe drivers for IPCs

11. Network transparent sockets

12. Network’s drivers for shared memory and Ethernet

13. RAM disk drivers for memory-resident files

14. Processor abstraction layer to enable application system design by user when using new versions of processor architecture.

Architecture:

VxWorks was initially a development and network environment for VRTX (Versatile Real-Time Executive). Later Wind River systems developed their own microkernel. So the VxWorks is of "client-server" architecture from the beginning. The heart of the VxWorks run-time system is the wind microkernel. This microkernel supports a full range of real-time features including multi-tasking, scheduling, inter task synchronization/communication and memory management. All the other functionality is implemented as processes.

VxWorks is highly scalable. By including or excluding various modules, VxWorks can be configured for the use in small embedded system with tough memory constraints to complex systems where more functions are needed. Furthermore, individual modules themselves are scalable. Individual functions may be removed from the library, or specific kernel synchronization objects may be omitted if they are not required by the application.

Task management:

The VxWorks real-time kernel provides a basic multitasking environment. VxWorks offers both POSIX and a proprietary scheduling mechanism (wind scheduling). Both pre-emptive priority and round-robin scheduling mechanism are available. The difference between POSIX and wind scheduling is that wind scheduling applies the scheduling algorithm on a system wide basis, whereas POSIX scheduling algorithms are applied on a process-by-process basis.IN VxWorks, the states encountered by the task are of 8 different types:

1. Suspended: idle state just after creation or state where execution is inhibited.
2. Ready: waiting for running and CPU access in case scheduled by the scheduler but not waiting for a message through IPC.
3. Pending: The task is blocked as it waits for a message from the IPC or from a resource; only then will the CPU be able to process further.
4. Delayed: sent to sleep for a certain time interval.
5. Delayed + suspended: delayed and then suspended if it is not pre-empted during the delay period.
6. Pended for an IPC [Inter process Communication] + suspended: Pended and then suspended if the blocked state does not change.
7. Pended for an IPC + delayed: Pended and then pre-empted after the delayed time interval.
8. Pended for an IPC + suspended + delayed: Pended and suspended after delayed time interval.

Kernel library functions are included in the header files ‘vxWorks.h’ and ‘kernelLib.h’. Task and system library functions are included in ‘taskLib.h’ and ‘sysLib.h’. User task priorities are between 101 and 255. Lowest priority means task of highest priority number (255). System tasks have the priorities from 0 to 99. For tasks, the highest priority is 100 by default.

The functions involved in task management:

1. Task Spawn function: It is used for creating and activating a task. Prototype is:

```
Unsigned int
taskId=taskSpawn(name,priority,options,stacksize,main,arg0,arg1,arg2
,….,arg9)
```

2. Task suspending and resuming functions:

`taskSuspend(taskId)`: inhibits the execution of task identified by taskId.

`taskResume(taskId)`: resumes the execution of the task identified by taskId.

`taskRestart(taskId)`: first terminates a task and then spawn again with its original assigned arguments.

3. Task deletion and deletion protection functions:

`taskDelete(taskId)`: this permanently inhibits the execution of the task identified by taskId and cancels the allocations of the memory block for the task stack and TCB.

Each task should execute the codes for the following:

1) Memory de-allocation
2) Ensure that the waiting task gets the desired IPC.
3) Close a file, which was opened before.
4) Delete child tasks when the parent task executes the exit( ) function.

4. Delaying a task to let a lower priority task get access: '`int sysClkRateGet ( )`' returns the frequency of the system ticks. Therefore to delay by 0.25 seconds , the function `taskDelay(sysClkRateGet ( )/4)` is used.

Memory management:

In VxWorks, all systems and all application tasks share the same address space. This means that faulty applications could accidentally access system resources and compromise the stability of the entire system. An optional tool named VxVMI is available that can be used to allow each task to have its own address space. Default physical page size used is 8KB. Virtual memory support is available with VxVMI tool. VxWorks does not offer privilege protection. The privilege level is always 0 (supervisor mode).

Interrupts:

To achieve the fastest possible response to external interrupts, interrupt service routines (ISRs) in VxWorks run in a special context outside of any thread's context, so that there are no thread context switches involved. The C function that the user attaches to a interrupt vector is not the actual ISR. Interrupts cannot directly vector to C functions.

The ISRs address is stored in the interrupt vector table and is called directly from the hardware. The ISR performs some initial work (e.g. saving registers and setting up stack) and then calls the C function that was attached by the user. For this reason, we use the term

interrupt handler (instead of ISR) to designate the user installed C handler function. VxWorks uses an ISR design that is different from a task design. The features of the ISR in VxWorks are:

1. ISRs have the highest priorities and can pre-empt any running task.
2. An ISR inhibits the execution of tasks till return.
3. An ISR does not execute like a task and does not have regular task context.
4. An ISR should not use mutex semaphore.
5. ISR should just write the required data at the memory or buffer.
6. ISR should not use floating-point functions as these take longer time to execute.

Performance:

- Real-time performance: Capable of dealing with the most demanding time constraints, VxWorks is a high-performance RTOS tuned for both determinism and responsiveness.
- Reliability: A high-reliability RTOS, VxWorks provides certification evidence required by strict security standards. Even for non-safety-critical systems, VxWorks is counted on to run forever, error free.
- Scalability: An indispensable RTOS foundation for very small-scale devices, large-scale networking systems, and everything in between, VxWorks is the first RTOS to provide full 64-bit processing to support the ever-growing data requirements for embedded real- time systems. VxWorks is scalable in terms of memory footprint and functionality so that it can be tuned as per the requirements of the project.
- Interrupt latencies: The time elapsed between the execution of the last instruction of the interrupted thread and the first instruction in the interrupt handler is Interrupt

latency. The time needed to go from the last instruction in the interrupt handler to the next task scheduled to run is Interrupt dispatch latency. VxWorks 5.3.1 exhibits an interrupt latency of 1.4 to 2.6 microseconds and a dispatch latency of 1.6 to 2.4 microseconds.

- Priority inheritance: VxWorks has a priority inheritance mechanism that exhibits an optimal performance, which is essential for an RTOS.
- Footprint: VxWorks has a completely configurable small memory footprint for today's memory-constrained systems. The user can control how much of the operating system he needs.

Applications:

VxWorks RTOS is widely used in the market, for a great variety of applications. Its reliability makes it a popular choice for safety critical applications. VxWorks has been successfully used in both military and civilian avionics, including the Apache Attack Helicopter, Boeing 787, 747-8and Airbus A400M. It is also used in on ground avionic systems such as in both civilian and military Radar stations. Another safety critical application that entrusts VxWorks is BMW's i-Drive system. However, VxWorks is also widely used in non-safety-critical applications where performance is at premium. The Xerox Phaser, a post-script printer is controlled by a VxWorks powered platform. Link Sys wireless routers use VxWorks for operating switches.

VxWorks has been used in several space applications. In Space crafts, where design challenges are greatly increased by the need of extremely low power consumption and lack of access to regular maintenance, VxWorks RTOS can be chosen as the operating system for On

Board Computer [OBC]. For example, ‘Clementine’ launched in 1994 is running VxWorks 5.1 on a MIPS-based CPU responsible for the Star Tracker and image processing algorithms. The ‘Spirit’ and ‘Opportunity’ Mars Exploration Rovers were installed with VxWorks. VxWorks is also used as operating system in several industrial robots and distributed control systems.

Conclusion:

The need to develop for real-time embedded applications is always a challenge, especially when expensive hardware is at risk. The complex nature of such systems requires many special de-sign considerations, an understanding of physical systems, and efficient management of limited re-sources. Perhaps one of the most difficult choices the embedded system designers have to make is which operating system they are going to use. It is critical to have operating system that will be able to be fail-safe, secure, scalable, fast, and robust in multitask management, while being friendly to the application developers. VxWorks is an RTOS which meets almost all these requirements.