VxWorks 6.0
Runtime Overview

Wind River TechDay / Eindhoven
6 Okt 04
Henk Schouten – Technical Account Manager
Agenda

• VxWorks 6.0 Key Focus
• VxWorks 6.0 Key Goals
• RTOS new Features
VxWorks 6.0 Key Focus

- Performance, Reliability and Productivity
- Seamless Migration from VxWorks 5.5
- State of the Art Memory Protection
  - Error detection and reporting
  - Real-Time Processes (RTP)
  - Inter-Process Communication (IPC)
- Modular Runtime and IDE
- Innovative Debugger: A System Solution
- Validated by Industry-leading Companies
## VxWorks 6.0 Timeline

<table>
<thead>
<tr>
<th>Event</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>VxWorks 6.0 Pre-Release 1</td>
<td>Feb 04</td>
</tr>
<tr>
<td>VxWorks 6.0 Pre-Release 2</td>
<td>Apr 04</td>
</tr>
<tr>
<td>VxWorks 6.0 Beta</td>
<td>Aug 04</td>
</tr>
<tr>
<td>VxWorks 6.0 FCS</td>
<td>Dec 04</td>
</tr>
</tbody>
</table>

### Protect
- Protects devices from errant code and malicious attacks
- Strong memory protection
- Memory error detection and reporting

### Connect
- Common message passing IPC
- Designed for future multi-processor, multi-core, multi-OS environments
- Location transparency

### Leverage
- Seamless compatibility with VxWorks 5.5
- Broad processor support
- Support for open source
- Extensible APIs
- Supported by Wind River Workbench
VxWorks 6.0

Key Goals
VxWorks 6.0 Key Goals

• Increase customer value by enabling:
  • Leverage and reuse of existing IP investment
  • Improved development experience and edit-compile-debug cycle
  • The ability to create more robust, higher quality end-products

• VxWorks 6.0 delivers on these goals by providing:
  • Easy migration from VxWorks 5.5
  • An optional, viable memory protection model
  • Continued focus on VxWorks key attributes
  • Best in class IDE (Eclipse based) and debugger (Multi-X) based on Wind River technology -> Wind River Workbench
**VxWorks 6.0 – Seamless Migration**

- **Seamless compatibility with VxWorks 5.5:**
  - recompile only
  - Runs existing VxWorks 5.5 BSPs, drivers and kernel applications

- **Easy migration of a user application from VxWorks 5.5 kernel to an RTP**

- **Extensive compatibility testing**
  - VxWorks 5.5 test suites
  - Customer code

**VxWorks 5.3.1**
- wind Kernel / Target Shell
- TCP/IP Networking
- Distributed Shared Memory

**VxWorks 5.4**
- Tornado 2 IDE
- Networking Routing
- File Systems
- Connectivity
- Graphics and Java

**VxWorks 5.5**
- Diab Compiler
- POSIX pThreads
- pSOS-Events

**VxWorks 6.0**
- Memory Protection
- Error Detection
- Real-Time Process
- IPC Messaging
- Unified MX Debugger
- Eclipse-based IDE
VxWorks 5.x Today

Shared Environment:
- Single address space
- Program code & data
- Kernel objects
- Heap
- Supervisor privilege tasks
- Direct device access
- Simple to understand
- Straightforward to program

But this has its challenges too…
- Resource reclamation difficult
- Symbols collide during integration
- App errors are unmanaged and so must be treated as system errors
- Software errors/faults can propagate
Technical Goals of VxWorks 6.0

- Reduce problematic error propagation
  - To help find root causes sooner

- Provide strong memory protection
  - To contain error propagation paths

- Provide message passing IPC: Wind River Multi Processing Technology
  - To facilitate the interaction of software separated by memory protection boundaries
Technical Goals of VxWorks 6.0

• Reduce problematic error propagation
  • To help find root causes sooner

• Provide strong memory protection
  • To contain error propagation paths

• Provide message passing IPC: Wind River Multi Processing Technology
  • To facilitate the interaction of software separated by memory protection boundaries
Memory Protection – Error Detection & Reporting

- Real-Time error detecting, reporting & handling
- Faster and cheaper development & testing cycles
- Increased robustness & reliability

Key features
- ISR/Task stack overrun detection
- Code corruption detection
- Null pointer usage detection
- Heap block overrun detection
- Heap usage tracking & leakage detection
- Persistent storage of error records
Error Detection (Using the MMU)

Stack Pages

Start

ISR/Task Stack Overrun Detection

End

Stack

Program Pages

Address Space

Read-Only Code

Code Corruption Detection

RW Data

Null Ptr Usage Detection

0x0
Heap Error Detection
(MMU not required)

Heap Blocks

Heap Block Overrun Detection

Heap Blocks

Task1
Task2
Task3

Heap Usage Tracking & Leakage Detection
Error Reporting Features

- Error records
- Error reporting & handling policy is customizable
- ISR contexts managed - not just tasks
- Persistent memory capability
Technical Goals of VxWorks 6.0

• Reduce problematic error propagation
  • To help find root causes sooner

• Provide strong memory protection
  • To contain error propagation paths

• Provide message passing IPC: Wind River Multi Processing Technology
  • To facilitate the interaction of software separated by memory protection boundaries
Memory Protection – Real-Time Processes (RTPs)

- Protects kernel from applications and applications from each other
- Familiar Unix®-style process programming model
  - Executing from a main() routine
- Easy integration and testing of legacy or third-party code
- MMU or MMU-less
- User-level objects (code, heap, stack, tasks, semaphores, FDs, etc.)
- Global task scheduler
- Resource reclamation

VxWorks Kernel Environment
VxWorks 5.5 Compatibility

- Goal is for VxWorks 5.5 BSPs, drivers, projects and kernel code to be directly importable without modification

- Migration story for legacy code is to “just run it in the kernel”

- Re-compilation required – source level compatibility only

- Some work must be done in order to benefit from strong memory protection (i.e. modifications are needed to RTP-ize software)

- POSIX PSE 51 + 52
Technical Goals of VxWorks 6.0

- Reduce problematic error propagation
  - To help find root causes sooner

- Provide strong memory protection
  - To contain error propagation paths

- Provide message passing IPC: Wind River Multi Processing Technology
  - To facilitate the interaction of software separated by memory protection boundaries
Memory Protection – Message Channels

- Common IPC mechanism for runtime integration of all applications and middleware
- New message passing model complements existing function call model
- Location transparency provides design flexibility
- Three messaging modes
  - Unidirectional send (“fire-and-forget”)  
  - Send with asynchronous reply  
  - Blocking send-receive-reply (RPC semantics)
- Designed for future multi-core and multi-processor messaging
VxWorks 6.0 Summary

• Backward compatibility with VxWorks 5.5
• State-of-the-art memory protection
• Memory error detection and reporting mechanisms
• Standard process-based programming model
• Scalable message passing architecture