Building Partitioned Architectures
based on the
Ravenscar Profile

Brian Dobbing
Chief Technical Consultant
Aonix Europe Limited, UK
brian@uk.aonix.com

SIGAda 2000 Laurel, November 2000
Agenda

- Software Partitioning
  - What is it?
  - Why is it needed for high integrity systems?

- Ravenscar Profile
  - What is it?
  - Why is it suited for high integrity systems?

- Can we use the Ravenscar Profile to implement a Partitioned System?
  - Example using the ARINC 653 Application Executive standard
Software Partitioning

- **Requirements to maximize processor usage**
  - But must ensure protection to preserve integrity

- **Within Safety Critical / High Integrity**
  - Spatial firewall to control access to code / data

- **Within Hard Real Time Systems**
  - Temporal firewall to guarantee time deadlines

- **Requirements for High Availability**
  - Replication to protect against faults
  - Dynamic re-configuration on node failure
Partitioning - single processor

- High Criticality
- Low Criticality
- Non-Critical

Controlled Communication

Firewalls

I/O devices
Physical memory access
Partitioning - distributed

- High Criticality
- Low Criticality
- Non-Critical

Hardware Nodes

Firewall

Kernel

I/O ports
Partitioning – high availability

Hardware Nodes

Firewall

High Criticality

Hot Standby

Non-Critical

Replicated Communication

Remote Communication

Kernel

Kernel
Partitioning System Example

- ARINC 653 Application Executive
  - Partitioning for Integrated Modular Avionics

  - Kernel is the “Module Operating System”
    - Controls the Module (usually the board)
    - Schedules the Application Partitions

  - Partition local control is via the “Partition Operating System”
    - Controls its Application
    - Schedules the Application Threads
The Kernel in APEX

The Kernel (Module Operating System):

- Is Highly Trusted Software
- Directly controls the Hardware (Supervisor mode)
- Manages the Spatial Firewalls using the MMU
- Manages the Temporal Firewalls using Fixed-Time-Slice Round-Robin Scheduling
- Controls Communication with other kernels via I/O ports
- Controls Detection of Faults, e.g. traps, and timeouts
- Supports Application Partitions written in any language via Standard API
The Partition in APEX

- The Application Partition:
  - Can use Internal Threads or be Sequential
  - Can be written in any Language using standard API
  - Contains Code / Data all at one Criticality Level
  - Usually Executes in User Mode
  - Executes within its own Address Space
  - Executes within its Fixed Time-Slice

- The Partition Operating System:
  - Schedules its Internal Threads
  - Implements Internal Thread Services
  - Interfaces to the Kernel for Inter-Partition Services
Example Module?

- Ada Flight Control System
- POS for Ada
- Ada RTS
- APEX Kernel (Module OS)
- H/W
- Java VM
  - In-flight entertainment
  - POS for Java
Critical Code in Ada95

- Ravenscar Profile – Ada tasking subset for...
  - Safety-Critical systems
    - Suitable for the most rigorous certification requirements
  - High Integrity systems (lower criticality than S-C)
    - Support functional static analysis and verification
  - Hard Real Time systems
    - Support temporal static analysis
    - Bounded Worst Case execution times and interrupt latency
  - Constrained embedded systems
    - Very small runtime footprint, very fast execution times
Ravenscar Profile (1)

- Major output from the 8th International Real-Time Ada Workshop
  - Held at Ravenscar, UK in April 1997
    - Defined in Ada Letters Vol 17 No 5 October 1997
    - See also www.cs.york.ac.uk/rts
    - See also paper in Proceedings of Ada-Europe 1998
  - Revised 9th/10th International Real-Time Workshops

- Included in an ISO Standard Technical Report
  - “Guide for the use of Ada in High Integrity Systems”
  - See ISO/IEC TR 15942:2000
  - Work in progress to consider how to add it to Ada0Y
What is in the Profile?

- Restricted *tasking* model for static timing analysis
  
  - All tasks created at startup, and run forever
  
  - Periodic task release via access to real-time clock device
  
  - Sporadic task release via event signalling or interrupt
  
  - Mutual exclusion via priority ceiling protocol emulation
  
  - Support for (nested) interrupt handlers
package RT renames Ada.Real_Time;

task body T1 is
    Current_Frame : RT.Time := T1_Start_Time;
    Frame : constant RT.Time_Span := RT.Microseconds (200);
begin
    loop
        delay until Current_Frame;
        -- Periodic processing code
        Current_Frame := Current_Frame + Frame;
    end loop;
end T1;

200 μsecs Frame

Current_Frame
package STC renames Ada.Synchronous_Task_Control;
package Sync is
  SO : STC.Suspension_Object;
  protected Trigger is
    procedure Int_Handler; -- Body performs Set_True(SO)
    pragma Attach_Handler (Int_Handler, Trap_Number);
  end Trigger;
end Sync;
task body T2 is begin
  loop
    STC.Suspend_Until_True (Sync.SO);
    -- Aperiodic processing code
  end loop;
end T2;

Aperiodic Task (no data)
Aperiodic Task (with data)

protected Trigger is

procedure Signal (D : in Data);
entry Wait (D : out Data);

private

Data_Store : Data;
Signal_Arrived : Boolean := False;
end Trigger;

task body T3 is

My_Data : Data;
begin

loop

Trigger.Wait (D => My_Data);

Process (My_Data); -- Aperiodic processing code
end loop;
end T3;
Priority Ceiling Locking

protected Resource is
    function Get return Data_Type;
    procedure Put (Latest : in Data_Type);
private
    Data : Data_Type := Null_Data;
    pragma Priority (40); -- Ceiling priority
end Resource;

Task 1
- priority 40
- priority 20

Task 2
- priority 40
- priority 30

Put

Get

Task 2 ready
Task Interaction Example

- H/w interrupt
- run

Interrupt Handler

- open
- Protected entry

Sporadic Task

- wait

Protected Object

- put

Periodic Task

- get
Profile Implementation

- Ravenscar Profile implemented by Aonix
  - "ObjectAda Raven™"
  - Ravenscar tasking subset plus ...
  - ... Deterministic sequential execution model
  - Open architecture for application-specific execution environments
  - Runtime System Certifiable to DO-178B Level A
    - Can map to other standards too e.g. Def-Stan 00-55
Implementing APEX

- Can we implement … :
  - The APEX kernel (MOS)
  - The Partition O/S (POS)
  - A Safety Critical Application Partition

- … using ObjectAda/Raven ?

- YES !
  - Otherwise I wouldn’t be doing this talk
Open Architecture

A Raven program can be configured for:

- **Internal Execution Behavior**
  - Startup / shutdown actions
  - Task scheduling at synchronization points
  - Storage management
  - Response to Exceptions
  - Task death and possible re-start

- **External Execution Environment**
  - Bare board or over an external executive / RTOS
Module Operating System(1)

The MOS ...

- Is non-tasking
- Is Interrupt-driven (timers, traps, I/O etc)
  - Requires pragma Attach_Handler
- Must initialize the hardware at start-up
  - SVC mode, MMU settings, trap handlers, etc
- Must perform partition context switching
- Must perform I/O to/from other modules
## Module Operating System (2)

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Raven</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ada95 interrupt handlers</td>
<td>☑</td>
</tr>
<tr>
<td>Initialize board (MMU, SVC mode)</td>
<td>☑ BSP</td>
</tr>
<tr>
<td>Assembler-level Context Switching</td>
<td>☑</td>
</tr>
<tr>
<td>Device-level I/O control</td>
<td>☑ BSP</td>
</tr>
<tr>
<td>Device-level Timer control</td>
<td>☑ BSP</td>
</tr>
<tr>
<td>Trap handling control</td>
<td>☑ BSP</td>
</tr>
</tbody>
</table>
Partition Operating System (1)

The POS …

- Supports multiple threads (static configuration)
  - Requires global tasks
- Supports Inter-Thread services (via Standard API)
  - Events, Blackboards, Buffers, …
  - Requires control over scheduling at synchronization points
  - Requires mutual exclusion, queuing, timeouts
- Supports Health Monitoring task
  - Must get control on trap, exception, time overrun etc
  - Requires task restart or partition restart (or ignore fault)
- Supports Inter-Partition Message Passing
  - Send message, Wait for message
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Raven</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library-Level tasks</td>
<td>✓</td>
</tr>
<tr>
<td>User-defined scheduling</td>
<td>✓</td>
</tr>
<tr>
<td>Shared resources</td>
<td>✓</td>
</tr>
<tr>
<td>Event management, queuing</td>
<td>✓</td>
</tr>
<tr>
<td>Exception handling, Task restart</td>
<td>✓</td>
</tr>
<tr>
<td>Timeouts, Partition restart</td>
<td>MOS</td>
</tr>
</tbody>
</table>
User-Defined Scheduling (1)

- APEX API used for all voluntary suspension
  - “I’m ready to start”
  - “I’ve finished my cycle”
  - “I’m waiting for an event”
  - “I’m waiting for a resource”
  - “I’m waiting for a message”

- POS handles pre-emption events directly

- Requirement is for POS to:
  - Suspend current task and to ready a new task
  - Achieved via per-task Suspension Objects
User-Defined Scheduling (2)

Threads = Ada tasks
Per-task Suspension Object

APEX API
Interrupt
Suspend_Until_True

Partition Operating System
Raven RTS

Set_True
Configure Execution Environment

- User-customizable Board Support Package
  - Board Interface
    - Machine level interface (for building a kernel in Raven)
    - Void when building an Ada user-mode Raven partition
  - Operating System Interface
    - Almost void for a bare machine kernel
    - Interface to kernel services for an Ada user-mode partition
  - Raven Instrumentation (RTS event handlers)
    - Almost void for a bare machine kernel
    - Response to events such as exceptions for user partitions
MOS / Ada-POS Construction

- The MOS can be constructed with its BSP
  - Run in supervisor mode
  - Control the board
  - Control the partitions

- The POS can be constructed with its BSP
  - Run in user mode
  - Control the threads
  - Interface to the MOS for external services
MOS Program Build

Certifiable

Predefined Library

Program Library

Board Library

System Library

Raven Library

Executable
Conclusions

- **Partitioning is very important requirement for**
  - Safety Critical
  - High Integrity
  - High Availability / Fault Tolerance

- **Ravenscar Profile targets these sectors**
  - Natural choice as the implementation environment

- **A COTS Implementation of the Profile can**
  - Implement a Partitioning System kernel
  - Implement an Operating System for an Ada Partition
  - Implement Ada Partition Application Code running over an external kernel
More Information

For more information
see www.aonix.com