Green Hills Software, Inc.

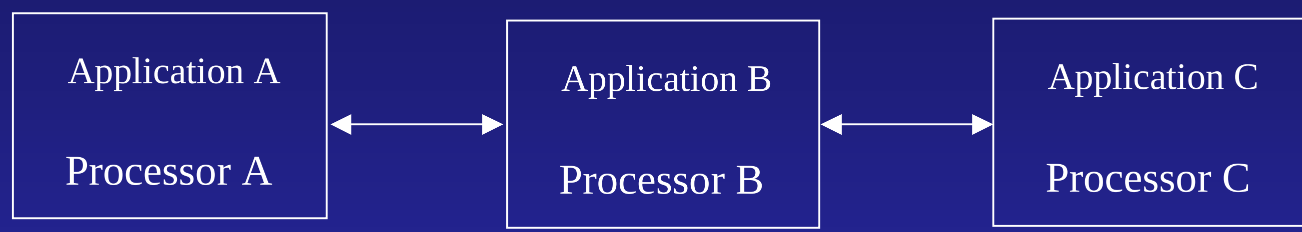
A Safe Tasking Approach to Ada95

Jim Gleason
Engineering Manager
Ada Products

Overview

- Multiple approaches to “safe” tasking with Ada95
  - No Tasking - SPARK
  - Ada95 Restricted Tasking Subset - Ravenscar
  - Non-Ada95 Tasking – ARINC653
- Tasking Environments
  - Bare Target – Ada runtime handles scheduling
  - Ada runtime on top of COTS Real-Time Operating System
- Safety-Critical RTOS Requirements
  - Partitioning (Time and Space)

Good "Olde" Days



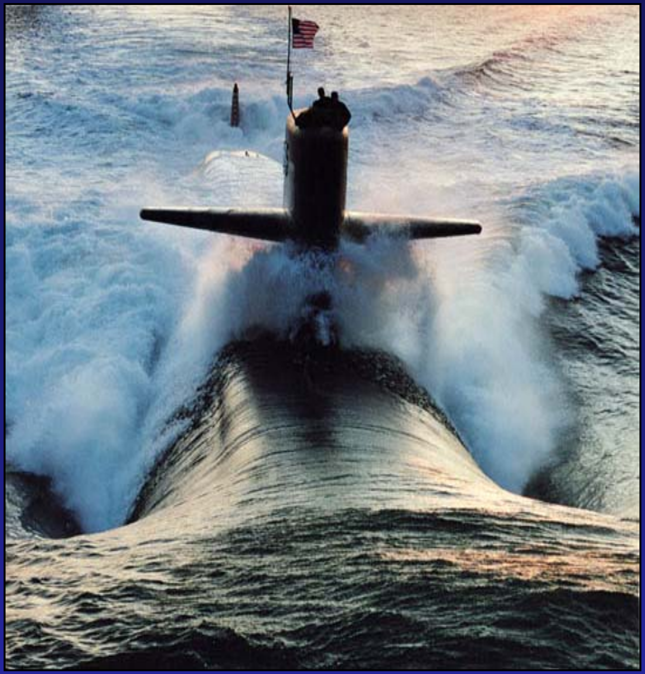
- Applications used no concurrent programming
- Rolled your own executive / run-time system (i.e. cyclic executive)
- Created life cycle data for library functions and executive
- Developed in-house tools to support debug and analysis needs
- Developed entire processor to same criticality level
- Primarily needed to pick a compiler vendor

Challenges Facing Embedded Systems Development Today

- Demand for new features
- Time to market
- Increasing complexity
- Component obsolescence
- Overall system costs
- Development assurance requirements
  - DO-178B, Common Criteria, OO-55, Internal

- Given these challenges, how can Ada95 be used to meet the needs of embedded applications?

Standard Ada Subsets



- **Defined by standards committees**
- **Safety emphasis**
- **SPARK subset (no Tasking)**
  - **Removed features that don't support Formal Program Verification**
    - **Tasking model**
    - **Gotos and arbitrary exits**
    - **Access types**
    - **Protected objects**
    - **Multi-level exception handling**
  - **Annotate program with formal specification of behavior**
    - **Tools (e.g. SPARK examiner) can verify program correctness**
  - **Temporal correctness is difficult to formally verify**

Ravenscar Profile

- Formed by the IRTAWG
- Accepted for inclusion in Ada 2005
- Deterministic features
  - Implemented by pragma Restrictions
  - Fixed set of tasks defined at the library level
  - Fixed set of protected objects defined at the library level
  - Synchronization objects (Ada.Synchronous_Task_Control)
  - No asynchronous transfer of control

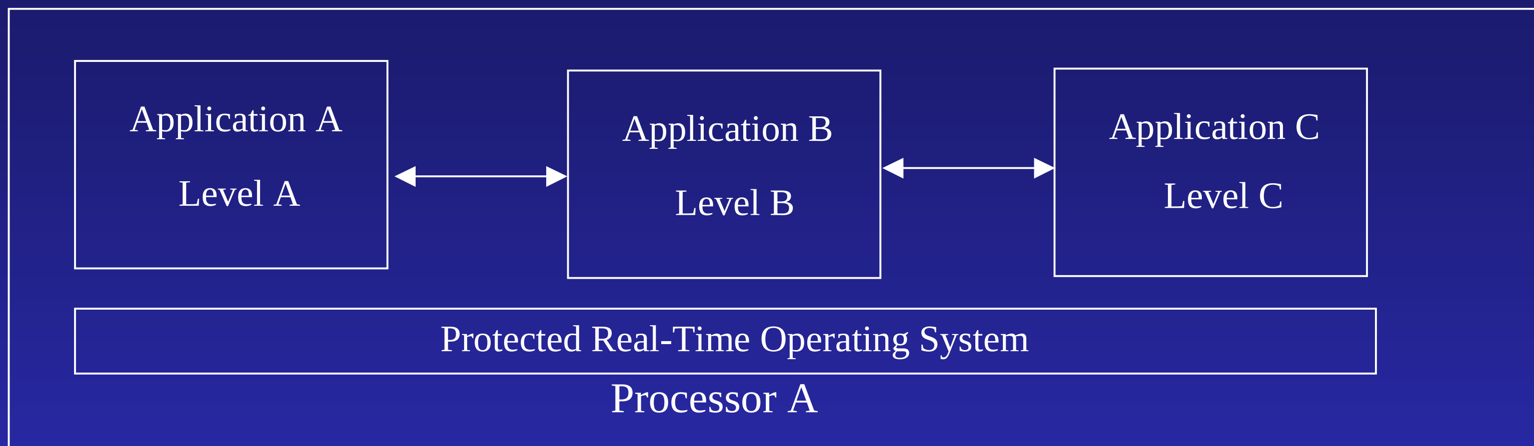
Ravenscar Profile

- **Deterministic features (cont.)**
  - **No dynamic memory allocation**
  - **No task entries and select statements (i.e. rendezvous)**
  - **No aborting of tasks**
  - **Only 1 entry with a single boolean per protected object**
  - **Only 1 task can wait on a barrier condition**
  - **No relative delays**
  - **Static Task Priorities**
  - **Protected Procedures as interrupt handlers**

ARINC653 **Application/Executive (APEX) interface**

- Defined for DO-178B Level A use
- Static creation / verification of system resources
  - Defined in configuration files
  - Tasks, Message Connections
- Time-based partition scheduling
  - Priority-based preemptive multi-tasking in partition
- Monitoring of Task deadlines
- Multi-Level error handling scheme (Module, Partition, Process)
- Message Passing Schemes
  - Sampling Port / Blackboard (Overwrite)
  - Queuing Port / Buffer (Buffered)
- Supplement 1 released in 2003
  - Focus is Portability
  - Defines standard configuration table format using XML

Today



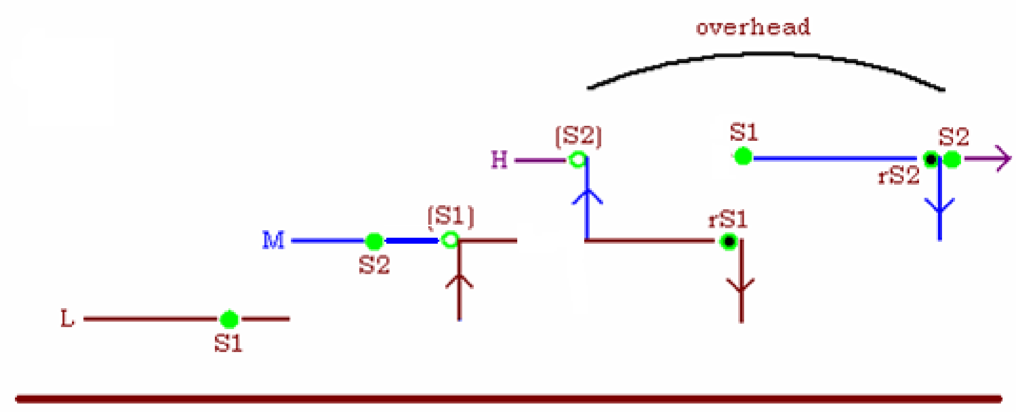
- **Applications developed using any one of the previously discussed tasking models**
- **Multiple applications running on a single processor**
- **Different criticality levels**
- **Need to leverage COTS for HW/SW obsolescence**
- **Vendor needs change dramatically**

Desirable Embedded RTOS Features

- Multi-tasking
- Safe semaphores (prevent priority inversion)
- Multiple virtual address space or partition support
- Inter-process communications
- Virtual and kernel mode device driver support
- One-shot and repeating alarms
- Shared memory with configurable access permissions
- Deterministic
- Small footprint but extendable for additional support
- Multi-language (Ada95, C, C++)
- Portability (Ada95, ARINC653, POSIX)
- Ported to multiple target processors

Priority Inheritance

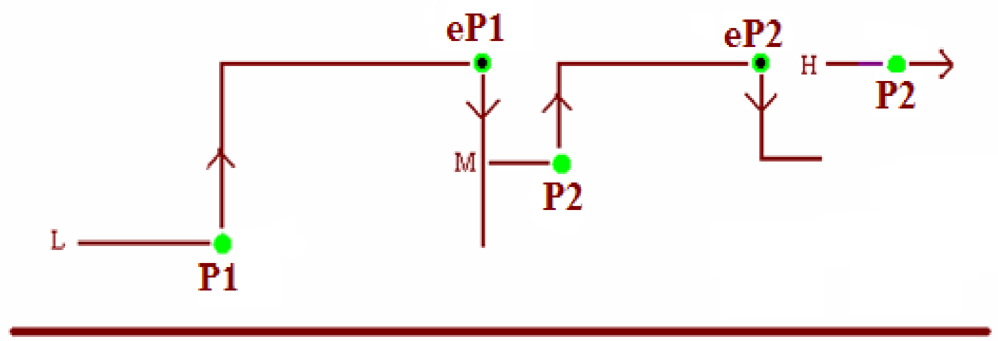
- Typical solution for priority inversion, priority inheritance semaphore, can cause chained blocking: complicates RMA and leads to an inefficient system
- Creates worst-case blocking times for high priority tasks



Chained Blocking Caused By Priority Inheritance

Priority Ceiling Protocol

- Priority ceiling protocol prevents priority inversion and chained blocking
- Protected object assigned a fixed priority – priority of the highest priority task that contends for the resource
- When task enters protected action, task priority is immediately elevated to protected object priority; task never blocks when executing protected procedure



Priority Ceiling Protocol

Multiple Virtual Address Spaces
Assure Availability and Integrity

Virtual Address Space 1

Ada95 Application

Safety Level : A (High)

Ada Run-Time

No Write Access

Virtual Address Space 2

Ada95 Application

Safety Level : C (Low)

Ada Run-Time

FAILURE!!!

NO EFFECT!

DO-178B RTOS (Protection)

Embedded Processor

Partitioning

- **Protection**
  - **Time and space allocated to one partition protected from corruption or use by another partition**
  - **Time and space of kernel protected from corruption or use by all partitions**
- **Benefits**
  - **Lower recertification costs, potentially**
  - **Typically small portion of application requires higher assurance level**
- **Protection in the Time Domain**
- **Protection in the Space Domain**
- **User Mode Device Drivers**

Partitioning Kernel Functional Requirements

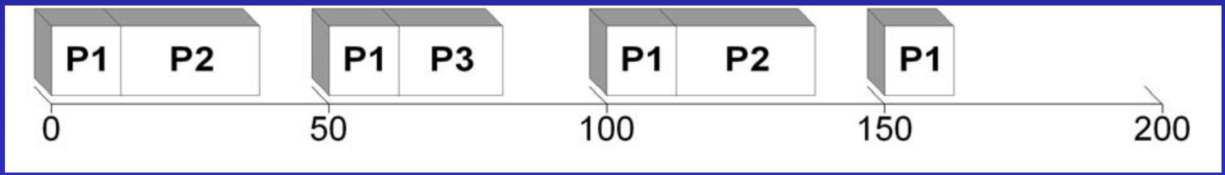
- **Micro Kernel Design**
  - **No file system**
  - **No built in communications stack**
  - **No shell**
  - **Static and dynamic scheduling**
  - **Static configuration files to prevent violations of security policies**
- **Kernel Extended by Middleware running in virtual address spaces**
  - **Secure file systems**
  - **Secure communications**
  - **Secure I/O**
  - **CORBA**

Protection in the Time Domain

- **Support for partition scheduling**
  - **Guaranteed time window to run**
- **Bounded time kernel calls**
  - **Long kernel calls checkpoint, allow preemption at API layer**
- **Non-preemptible kernel, no semaphores (priority inversion)**
- **“Hard Currency” OS – Address Spaces own Memory Regions and donate them to kernel, for kernel calls (i.e. no dynamic memory allocation done by kernel)**
- **Pure software timers with access permissions**
- **No hidden execution time / latency**
  - **Message transfers use task’s execution time**
  - **Never disable interrupts to update kernel structures**
- **Highest Locker (Priority Ceiling) semaphores**
  - **Absence creates worst-case blocking times for high priority tasks**

Partition Scheduler

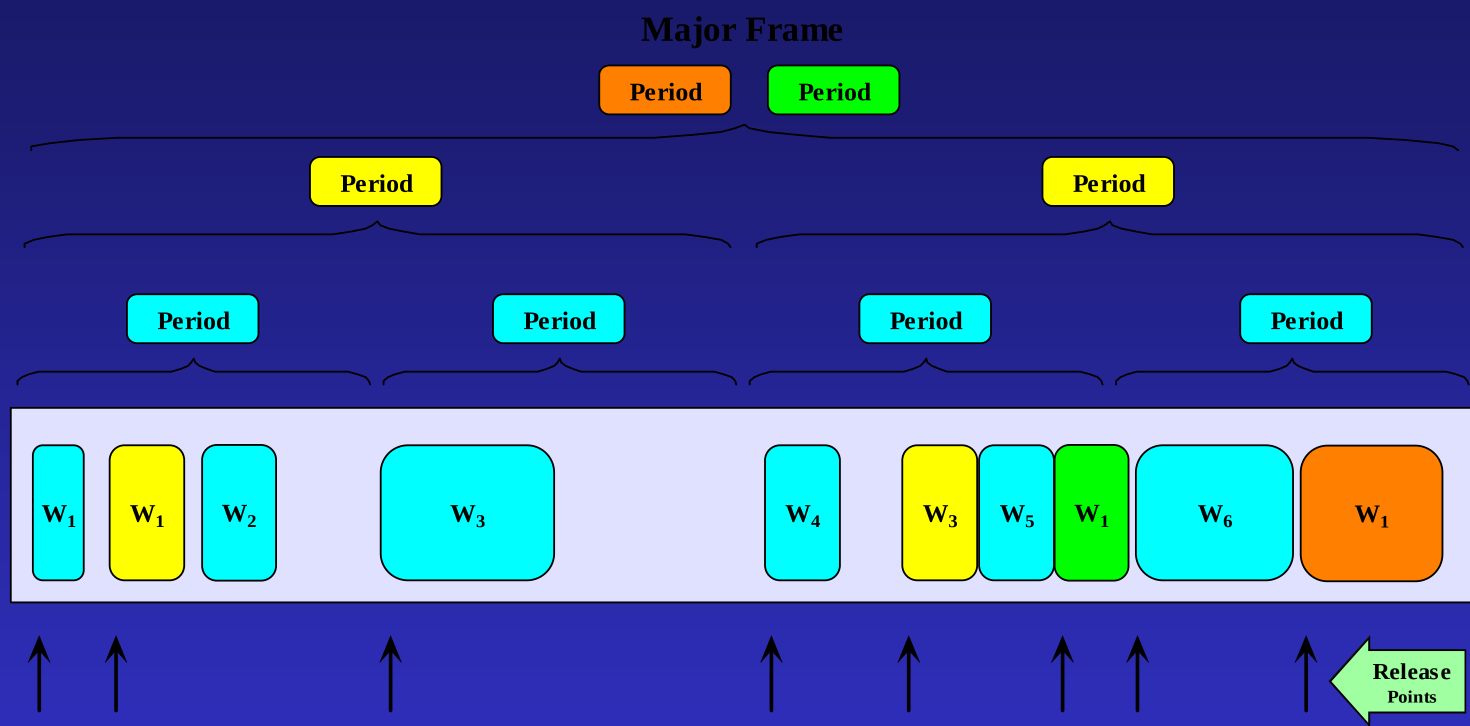
Partition	Offset	Execution Time
P1	0	10
P1	50	10
P1	100	10
P1	150	10
P2	10	20
P2	110	20
P3	60	15

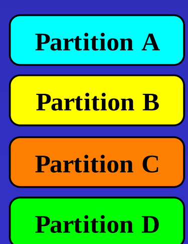


{Major Frame Period = 200}

- Normal preemptive priority-based task scheduler within a partition.
- Background priority tasks can run during non-allocated time.

ARINC653 Partition Scheduler





Note: Applicable to starting processes following SET_PARTITION_MODE to NORMAL or initiating START or DELAYED_START after the operating mode is NORMAL. Once started, the period of the processes defines when they will run. PERIODIC_WAIT does not depend on the partition release points.

- **Priority preemptive scheduling within partition**

Protection in the Space Domain

- No use of kernel addresses, all kernel call arguments verified before use, all object pointers tagged when object is freed
- Use of hardware memory management units
- Guaranteed resource availability
  - Address Spaces donate own memory to satisfy request
  - Address Space’s memory is completely protected from any other Address Space
  - Kernel Objects stored in kernel pages owned by the Address Space, completely protected
- Statically verifiable MMU settings
  - No dynamic manipulation of MMU to support message passing
  - Strict copy from one Address Space’s memory to another
- Connections – secure inter-address space communication
- Secure Device Drivers – User Mode tasks which use Connections as interface to the interrupt service routine
- “Code” as well as Data protected by MMU

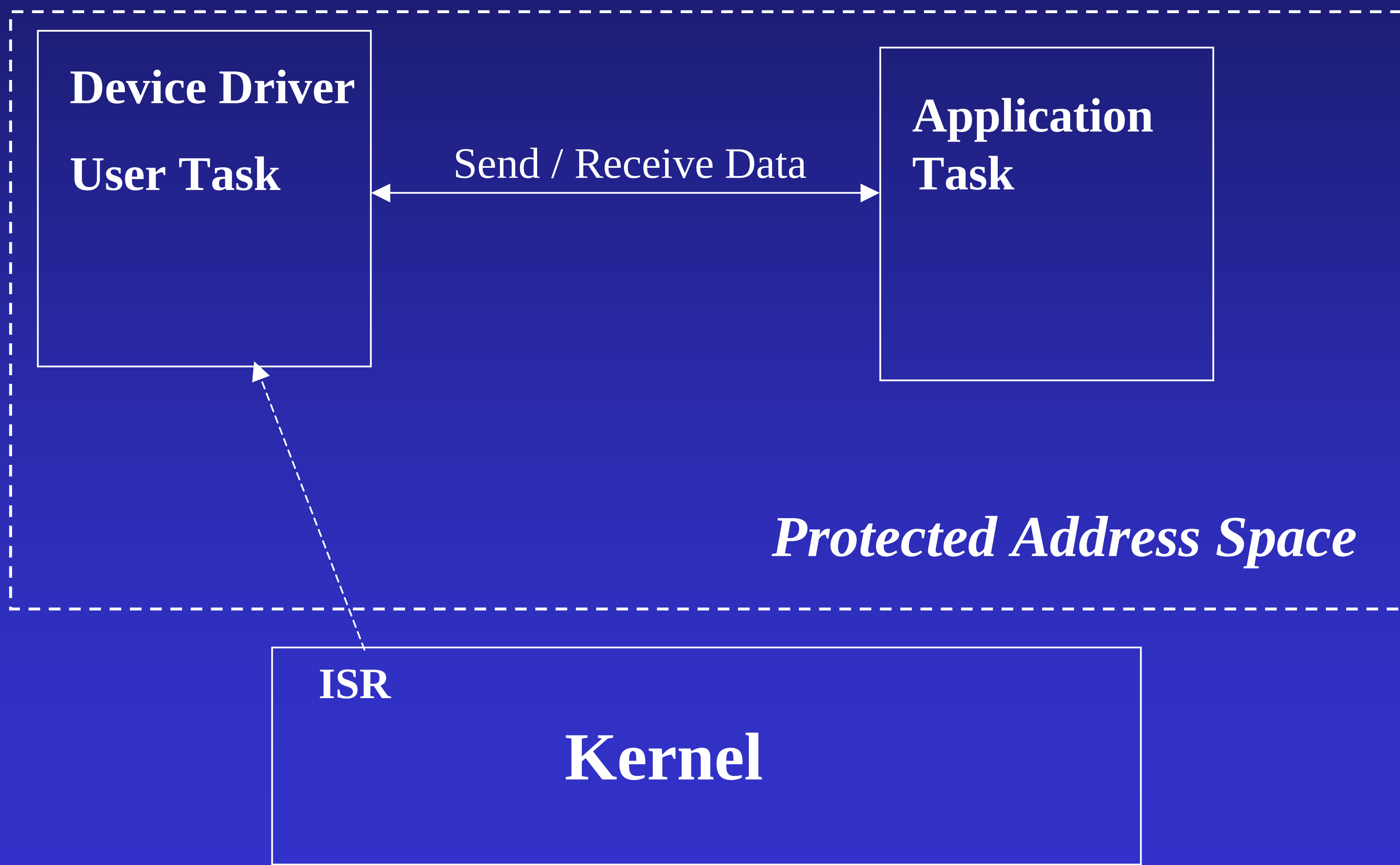
Protection in the Space Domain

- **Memory Protection (Data Isolation)**
  - **User Tasks in One AddressSpace Cannot Corrupt (neither maliciously or accidentally) Another AddressSpace or the Kernel**
  - **Utilizes MMU Hardware**
  - **Tasks Attempting Violation are Halted by Kernel**
- **Secure Services**
  - **All Kernel Service calls in Virtual AddressSpaces Occur via Trap mechanism where objects are represented by small integers.**
  - **Kernel controls all Object tables and verifies that all arguments to the kernel calls are valid.**
  - **Kernel Space Protected From Access by Virtual AddressSpaces**
- **Guaranteed Resource Availability (Denial of Service Prevention)**
  - **Each AddressSpace has a quota of memory set up statically**
  - **One AddressSpace cannot exhaust or affect the memory availability of another AddressSpace**

User Mode Device Drivers

- **User Mode Task**
  - **Runs in its own virtual address space**
  - **Does nearly all of the work**
- **Memory mapped I/O**
- **Interrupt Service Routine**
  - **Short**
  - **Service the physical device interrupt**
- **Completely secure**
  - **Can’t corrupt OS or any other AddressSpace or device**
  - **Can’t be corrupted by any other AddressSpace or device**
- **Written in a high level language**
  - **Easier to program**
  - **Easier to debug**
  - **Runtime error detection (e.g. stack overflow)**
  - **Performance analysis**
- **Fully preemptible: creates no interrupt shadow**

User Mode Device Driver



Core Processor Architecture

Virtual Address Space 1

Ada or C/C++ Application

Safety Level : A (High)

SPARK Ada Run-Time

C/C++ Run-Time

ARINC653 Library

NO EFFECT!

Virtual Address Space 2

Ada or C/C++ Application

Safety Level : C (Low)

Ravenscar Ada Run-Time

C/C++ Run-Time

ARINC653 Library

FAILURE!!!

RTOS (Protection)

Embedded Processor

Integrated Solution

Compilers
- Multi-Language
- Validated
- Multi-Processor

Run-Time Systems
- Safe Subsets
- Robust Partitioning
- Multi-Processor

Debug/Analysis Tools
- Many connections
- Run-Time System Aware
- MCDC support

Application A
Level A

Application B
Level B

Application C
Level C

Protected Real-Time Operating System

Processor A

- Run-Time System DO-178B Level A Life Cycle Data
- MCDC Tool Qualification Data
- Robust Compiler Technology
- Extensive debug capability

} Long Life-Cycle Support

Key Points

- **Single vendor solution**
  - **Tight integration of Ada95 compiler, Ada95 run-time system and RTOS**
  - **Clean certification package for total solution**
  - **One-stop technical support**
- **Green Hills produces products to support all software architectures**
  - **GMART – SPARK based Ada run-time supporting bare target, INTEGRITY and INTEGRITY-178B**
  - **GSTART – Ravenscar based Ada run-time supporting bare target, INTEGRITY and INTEGRITY-178B**
  - **INTEGRITY and INTEGRITY-178B – Partitioned (time and space) real-time operating system**
  - **ARINC653 application/executive compliant library**

INTEGRITY-178B