Real-Time Systems Programming with GNAT and OpenRavenscar

Juan Antonio de la Puente
<jpuente@dit.upm.es>
DIT/UPM, Technical University of Madrid
Overview

◆ High Integrity Systems and the Ravenscar profile
◆ The OpenRavenscar project
◆ The OpenRavenscar kernel
  – Integration with GNAT
  – Kernel interface
  – Design issues
  – Other components
◆ The OpenRavenscar tool set
◆ Developing real-time systems with OpenRavenscar
◆ Lessons learned and future work
High Integrity Systems

- Systems with strong safety and reliability requirements
  - static and dynamic analysis required according to different certification standards
- Ada is the language of choice for HIS
  - careful design
  - annex H: Safety and security
- A *safe subset* of the language is often used
  - traditional approach: no tasking (e.g. SPARK)
  - more recently: safe tasking profile (Ravenscar)
The Ravenscar Profile

- Ada tasking subset for high-integrity applications
- Defined at IRTAW 8 (Ravenscar, UK, 1997)
- Revised at
  - IRTAW 9 (Tallahassee, Florida, 1999)
  - IRTAW 10 (Las Navas del Marqués, Spain, 2000)
- Strategy
  - remove constructs with
    » high overhead
    » non-predictable behaviour
  - allow
    » timing analysis
    » small, fast, reliable runtime system
Ravenscar tasking model

- Static tasks and protected types/objects at the library level
- Protected objects with at most one entry with a simple barrier
  - no more than one task queued on an entry
- Synchronous task control
- Real-time package and delay until statement
- FIFO within priorities and ceiling locking scheduling
- Protected procedure interrupt handlers

Model enforced by pragma restrictions at compile time
(except for task termination and entry queue)
Forbidden features

- Task hierarchies
- Protected object hierarchies
- Dynamic POs and tasks
- Task entries
- Protected types with more than one entry
- Complex barriers
- More than one task in one entry queue
- Requeue
- ATC
- Select statement
- Abort
- Dynamic priorities
- Calendar package
- Relative delays
- Asynchronous task control
- User-defined task attributes

Supported features

- Library level tasks and POs
- Task discriminants
- Task identifiers
- FIFO within priority and Ceiling Locking policies
- Real-Time package
- Delay until statements
- Protected procedures as interrupt handlers
- Synchronous task control
- Atomic and Volatile pragmas
- Count Attribute (but not within entry barriers)
# Ravenscar profile restrictions

<table>
<thead>
<tr>
<th>Standard in Ada 95</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>No_Task_Hierarchy</td>
<td>Simple_Barrier_Variables</td>
</tr>
<tr>
<td>No_Abort_Statements</td>
<td>Max_Entry_Queue_Depth =&gt; 1</td>
</tr>
<tr>
<td>No_Task_Allocators</td>
<td>No_Calendar</td>
</tr>
<tr>
<td>No_Dynamic_Priorities</td>
<td>No_Relative_Delay</td>
</tr>
<tr>
<td>No_Asynchronous_Control</td>
<td>No_Protected_Type_Allocators</td>
</tr>
<tr>
<td>Max_Task_Entries =&gt; 0</td>
<td>No_Local_Protected_Objects</td>
</tr>
<tr>
<td>Max_Protected_Entries =&gt; 1</td>
<td>No_Requeue</td>
</tr>
<tr>
<td>Max_Asynchronous_Select_Nesting =&gt; 0</td>
<td>No_Select_Statements</td>
</tr>
<tr>
<td>Max_Tasks =&gt; N</td>
<td>No_Task_Attributes</td>
</tr>
<tr>
<td></td>
<td>No_Task_Termination</td>
</tr>
</tbody>
</table>
The Open Ravenscar project

- Ravenscar profile-compliant runtime for GNAT
  - GNAT based cross-compilation system for ERC-32 (SPARC V7)
  - distributed as free software (GPL)

- Launched and funded by ESA/ESTEC
  - programme on software development tools for ERC32
    » commercial tools
    » open-source tools

- Time frame
  - Phase 1: October 1999 - June 2000   GNAT/ORK 2.1
  - Phase 2: November 2000-May 2001
The ORK consortium

- ESA/ESTEC
  Requirements, reviews, acceptance

- University of York RTSG Consulting

- DIT/UPM
  ORK implementation

- EADS/CASA Space Division
  Validation & Verification

- Universidad Rey Juan Carlos - GSYC
  Debugger adaptation
ORK inputs

- **GNARL**
  - running on pthreads
  - Ravenscar restricted tasking selected by a pragma

- **JTK (José’s Tasking Kernel)**
  - internal DIT/UPM development
  - running on bare ix86
  - full Ada tasking with GNAT
GNAT support for the Ravenscar Profile

◆ All RP restrictions implemented
  – compile-time checking possible for most of them
◆ Standard RTS (GNARL & GNULL) not adapted to RP
  – dynamic storage, task entries, etc. in RTS
  – complex dependencies between RTS packages
◆ Special pragma Ravenscar
  – selects all RP restrictions (and two additional ones)
  – selects a restricted RTS
    » simplified tasking support
    » no support for interrupt handlers
◆ Little support for cross-compilation
  – special versions of libraries required (newlib)
GNAT Run-Time Architecture

Runtime system

Ada application
GNARL
GNUILI
pthread layer
operating system
hardware

Replace by
RT kernel
Initial Open Ravenscar Architecture

new components

- RP-compliant Ada application
- restricted GNARL
- adapted GNULI
- real-time kernel
- hardware
The Open Ravenscar Real-Time Kernel (ORK)

- A special-purpose real-time kernel providing support for RP-compliant programs compiled with GNAT
  - initial implementation for ERC-32
- No need for pthreads interface
  - pthreads duplicate much of the Ada 95 tasking functionality
  - almost direct implementation of GNULL interface possible
- Small, robust, efficient implementation possible
  - opens the way to HIS certification
Design decisions (1)

- Code in Ada as far as possible
  - safe sequential Ada subset used
- Keep GNARL unmodified
  - but some GNARL packages had to be changed
- Take advantage of RP restrictions to simplify the implementation
- Careful implementation of kernel primitives to provide effective timing analysis
Design decisions (2)

- Allow implementation of device drivers at application level
  - GNARL interrupt support had to be redesigned
- Effective real-time support
  - fine-granularity clock
- Isolate target dependencies to allow easy retargeting
- Include configuration support for hardware and kernel parameters
Actual Open Ravenscar Architecture

- RP-compliant Ada application
- restricted GNARL
- adapted GNULL
- real-time kernel
- hardware
GNARL / ORK interfaces

Ada.Synchronous_Task_Control
Ada.Interrupts.Names
System.Interrupts
System.Tasking.Protected_Objects.Single_Entry
System.Interrupt_Management
System.Interrupt_Management.Operations
System.Task_Primitives.
Operations
System.Task_Primitives
System.OS_Interface
System.OS_Primitives

GNULL Interface

Kernel Interface

Other GNARL packages

GNARL / ORK interfaces
ORK functionality

- Task management
- Task synchronization
- Scheduling
- Storage allocation
- Time-keeping and delays
- Interrupt handling
ORK architecture
Thread management

- Threads are created **only at start-up**
  - storage management simplified

- **Threads do not terminate**
  - user-defined procedure called if a thread attempts to terminate
  - default: raise Program_Error

- **FIFO within priorities dispatching**
  - ready queue: priority-ordered double linked queue

- **Thread-safe kernel**
  - monolithic monitor, interrupts disabled
  - no separate kernel and user modes
Thread synchronization

◆ Mutexes
  – ceiling locking (immediate ceiling priority inheritance)
  – direct implementation of mutual exclusion
    » by raising priority to ceiling priority
  – need to detect calls to potentially blocking operations
    » raise Program_Error

◆ Condition variables
  – condition variables with at most one waiting thread
    » raise Program_Error or user-defined procedure
  – no queues required

◆ Almost direct implementation of GNUILLI operations
  – no need for all pthreads complexity
Storage management

- Provides initial storage allocation for TCBs and stacks
  - pre-allocated TCB space
  - task space based on pragma Storage_Size
  - no storage allocation after start-up
  - no de-allocation

- Application code does not use storage pools
  - problems with some GNARL packages

- Thread stack limits are protected for improved safety
  - forbidden blocks between adjacent stacks
Interrupt handlers (1)

- **Only protected handlers supported**
  - direct attachment of hardware interrupts to protected handlers
  - no need for POSIX signals

- **Interrupt support in GNARL redesigned**
  - simpler implementation
  - no service tasks
  - special interrupt stack
Interrupt handlers (2)

original GNARL

modified GNARL/ORK
Time management

◆ Monotonic clock
  – integer nanoseconds from system start-up
  – high resolution tick
    » equal to the hardware clock period (100 ns on 10MHz ERC32)
  – efficient timekeeping
    » clock interrupts only every 1s

◆ Efficient absolute delays
  – alarm-clock model
  – delayed queue implemented as a single-linked list ordered by wake-up time
  – no cancellation
  – all delayed tasks made ready at once
High-resolution tick

clock pulses → Scaler → Control

zero indication → Counter

Set Preload

interrupt
The Open Ravenscar tool set (1)

Cross-Compilation System

- Based on GNAT 3.13
- Development platform: PC GNU/Linux
  - Solaris coming soon
- Execution platform
  - ERC32-based computer
  - TSIM ERC32 simulator
- Components
  - sparc-ork-gcc, sparc-ork-gnatbind, sparc-ork-gnatlink
  - sparc-ork-gnatmake
  - other gnat & binutils tools
The Open Ravenscar tool set (2)

**Debugging tools**
- GDB 4.17 & DDD 3.2 for GNU/Linux
  - scripts added for enhanced tasking support

**TSIM Simulator**
- Not part of Open Ravenscar
  - distributed by ESA/ESTEC
- Allows debugging on development platform before going to the actual target hardware
Developing with Open Ravenscar

- Write a Ravenscar profile-compliant program
  - a special gnat.adc file is required
- Compile and link with sparc-ork-gnat
  - most RP violations checked at compile-time
  - result is an ELF executable file
- Debug the program
  - on the development platform:
    » TSIM and GDB/DDD
  - on the target platform:
    » download and remote debugging monitor (not yet available)
- Make a PROM
  » tools not yet available
GNAT/Open Ravenscar compilation process

- gnat.adc
- application sources
- RTS specs
- application ALI files
- application object files
- RTS & library object files
- GNAT binder
- elaboration code
- GNAT linker
- ELF-32 SPARC executable

GNAT compiler
GNAT configuration file

-- file gnat.adc

pragma Ravenscar;

pragma Restrictions (Max Tasks => N);
-- N must be equal to the number of tasks of the
-- application

-- other restrictions can be declared here

pragma Task Dispatching Policy (FIFO Within Priorities);
pragma Locking Policy (Ceiling Locking);
Compile time and run time checks

- Most Ravenscar Profile restrictions checked at compile time

- Two restrictions checked at run time
  - No_Task_Termination
    » call a user-defined procedure (default silent)
  - Max_Entry_Queue_Depth => 1
    » raise Program_Error
Configuring ORK

◆ Configurable parameters in Kernel.Parameters
  – Maximum number of threads
  – Amount of memory available in the target board
  – Amount of memory space reserved for thread stacks
  – Size of the interrupt stack
  – Priority range
  – Clock frequency
  – Clock interrupt period

◆ Interrupt names and other parameters defined in System.OS_Interface (and Ada.Interrupts.Names)

◆ Requires re-building the kernel from sources
Kernel metrics
(ERC32 at 10 MHz)

- Size of kernel sources (statements)
  - 1361 Ada statements
  - 478 assembly code
  - 82 C

- Size of kernel binary code
  - 15 KB (with vector table)

- Minimum program size
  - 94 KB (no tasks)
  - 133 KB (with tasks)

- Context switch time
  - 85 μs

- Interrupt latency
  - 295 μs
Conclusions

◆ It is feasible to build new, simpler kernels for GNAT
  – GNULL concept and interface very useful
  – but reducing the memory footprint requires rewriting most of GNARL
  – it is possible to use a safe sequential subset of Ada for the kernel itself

◆ Certification of applications will require more effort
  – especially in upper GNARL layers
Current and future work

- Porting a reusable component framework (OBOSS) to GNAT/ORK
- Test on real hardware
  - board monitor & loader to be written
- Integrate in ESA Software Engineering Environment
  - generate RP-compliant code from HRT-HOOD designs
  - integrate timing analysis tools
- Improve integration with GNAT and prune GNARL of unused code
The ORK team

**ORK development** (DIT/UPM)
- Juan Antonio de la Puente
- Juan Zamorano
- José Ruiz
- Ramón Fernández
- Rodrigo García

**ORK validation** (EADS CASA)
- Jesús Borruel
- Juan Carlos Morcuende

**GDB/DDD adaptation** (URJC)
- Jesús González-Barahona
- Vicente Matellán
- Andrés Arias
- Juan Manuel Dodero

**RP consultants** (U York)
- Andy Wellings
- Alan Burns
Availability

source and binaries for PC GNU/Linux available on

http:www.openravenscar.org

current version is 2.1