CIFO 3.0

Pat Rogers

ABSTRACT

The Ada Runtime Environment Working Group (ARTEWG) has, since 1985, developed and published the Catalog of Interface Features and Options (CIFO) for Ada runtime environments. These interfaces, expressed in legal Ada, provide "hooks" into the runtime system to export both functionality and enhanced performance beyond that of "vanilla" Ada implementations. Such enhancements include high- and low-level scheduling control, asynchronous communications facilities, predictable storage management facilities, and fast interrupt response. CIFO 3.0 represents the latest release, which incorporates the efforts of the European realtime community as well as new interfaces and expansions of previous catalog entries. This presentation will give both an overview of the Catalog's contents and an "insider's" view of the Catalog as a whole.

BIOGRAPHY

Pat Rogers is a Consulting Scientist at SBS Engineering in Houston, where he is the principal investigator for a project which has developed a Distributed Ada implementation for the U.S. Air Force. He has been involved with Ada since 1980, is a founding member of ARTEWG, and is a contributing member of the CIFO development subgroup.
The New CIFO: Bridging Ada83, Realtime Systems and Ada9X

Pat Rogers
SBS Engineering, Inc.
18333 Egret Bay Blvd., Suite 340
Houston, Tx 77058
713/333-5040

What Is CIFO?

- Catalog of Interface Features and Options
- A set of common packages, subprograms and pragmas used to extend the capabilities of the Ada baseline facilities (the RM) via the runtime environment
- Developed by the Ada Runtime Environment Working Group -- ARTEWG ("art-wig")
- Began in 1985, first working meeting at UHCL
  - Dr. Charles McKay is chair of team
- Many Ada compiler vendors supporting CIFO
- Many users, including Space Station Freedom contractors
The Extended Runtime Library

Structural Differences From 2.1

- Reorganized Entry groupings
- New sections per Entry
  - Interactions With Other Entries
  - Changes From The Previous Release
- Interactions Matrix
Convergence with ExTRA

- Extensions des services Temps Reel Ada
  - Ada Real Time Service Extensions
- Many Entries added
- A few Entries superseded
- Finalized the change in CIFO orientation

Change In Orientation

- Early CIFO releases
  - Targeted to hard-constrained applications
  - Open to abuses and misuses
- Later CIFO releases
  - Less specific to hard-constrained applications
  - Abuses covered, but at what price?
Functional Categories

- Basic Mechanisms
- Scheduling Controls
- Asynchronous Cooperation Mechanisms
- Interrupt Support
- Compiler Directives
- Memory Management Mechanisms

*Indicates a new entry in following pages

Basic Mechanisms

- Task Identifiers
- Queuing Discipline
  - Provides basic definitions for task entry and scheduling disciplines
Scheduling Control

- Synchronous & Asynchronous Scheduling
  - Priority Inheritance Discipline
    - Provides priority inheritance configuration in the RTS
- Dynamic Priorities
- Time Critical Sections
- Abort Via Task Identifier
- Time Slicing

Scheduling Control

- Task Suspension
  - Provides a low-level means of controlling dispatching and execution, in a cooperative manner
- Two-Stage Task Suspension
  - Provides low-level controls that avoid race conditions
- Asynchronous Task Suspension
  - Allows one task to bilaterally prevent execution of another
- Synchronization Discipline
  - Allows specification of criteria for queuing entry calls and choosing among open select alternatives
Asynchronous Cooperation

- Resources
  - Provides efficient access control for (hardware) resources
- Events
  - Provides efficient task notification of latched conditions
- Pulses
  - Provides efficient task notification of non-latched conditions
- Buffers
  - Provides efficient asynchronous intertask communication
- Blackboards
  - Provides efficient inter-task messages

Asynchronous Cooperation

- Mutually Exclusive Access to Shared Data
- Broadcasts
  - Provides an efficient message broadcast capability
- Barriers
  - Allows simultaneous resumption of a fixed number of waiting tasks
- Asynchronous Transfer of Control
  - Supports ATC for fault recovery, mode changes etc.
- Shared Locks
  - Provides a very sophisticated lock facility
- Signals
  - Supersedes previous "Asynchronous Entry Call" interface
Interrupt Support

- Interrupt Management
- Trivial Entries
- Fast Interrupt Pragmas

Compiler Directives

- Pre-elaboration of Program Units
- Access Values That Designate Static Objects
  - Provides a more portable means to reference static objects
- Passive Task Pragmas
  - Provides a standardized approach to task "passification"
- Unchecked Subprogram Invocation
  - Provides more reliable means of invocation by address
- Data Synchronization Pragma
  - Allows CIFO task synchronization facilities to be used to share data
Memory Management

- Dynamic Storage Management
  - Provides predictable, flexible storage management facility

Deleted Entries

- Special Delays
  - One of the vendors convinced us that it was counter-productive

- Transmitting Task Identifiers Between Tasks
  - Removed in lieu of a more safe, coordinated approach to distribution facilities
Priority Inheritance

package Priority_Inheritance_Discipline is
    procedure Set_Priority_Inheritance_Criteria;
    procedure Reset_Priority_Inheritance_Criteria;
end Priority_Inheritance_Discipline;

pragma Set_Priority_Inheritance_Criteria;

- Enables/disables priority inheritance
- Procedural interface allows switching !!

Task Suspension

with Task_IDs;
package Task_Suspension is
    procedure Enable_Dispatching;
    procedure Disable_Dispatching;
    Function Dispatching_Enabled return Boolean;
    procedure Suspend_Self;
    procedure Resume_Task( Target : in Task_IDs.Task_Id );
end Task_Suspension;

- Tasks can control their own suspension
- Safe if not multiprocessing
Two-Stage Task Suspension

with Task_IDs;
package Two_Stage_Task_Suspension is
    Suspension_Error: exception;
    procedure Will_Suspend;
    procedure Suspend_Self;
    procedure Resume_Task( Target: in Task_IDs.Task_Id );
end Two_Stage_Task_Suspension;

• Safe for multiprocessing

Asynchronous Task Suspension

with Task_IDs;
package Asynchronous_Task_Holding is
    procedure Enable_Holding;
    procedure Disable_Holding;
    function Holding_Enabled return Boolean;
    procedure Hold_Task( T: in Task_IDs.Task_Id );
    procedure Hold_Task( T: in Task_IDs.Task_Id; Held: out Boolean );
    procedure Release_Task( Target: om Task_IDs.Task_Id );
end Asynchronous_Task_Holding;

• Controversial, but considered necessary
  • All three Entries designed to interact predictably
Designating Static Objects

generic
  type Object is limited private;
  type Reference is access Object;
function Make_Access_Value( Static : Object ) return Reference;
pragma May_Make_Access_Value( <type_mark> );
Make_Access_Supported : constant Boolean := <value>;

CIFO Procurement Issues

- "More" is not "Better"
  - Unused Entries slow down others
  - Some Entries will never be implemented
  - Some Entries will conflict with others
- Conformance is only per Entry
  - Semantics of Entry
  - Interactions with other Entries
Adult Programming

- Don't mix conflicting Entries
  - Example: Various scheduling controls
  - Use the Interactions Matrix
  - Some Semantic ambiguities may still exist
- Don't use the "dubious" Entries
  - The procedural interface to Priority Inheritance Discipline (use the pragma)

CIFO and Ada9X

- Should 9X obviate a CIFO?
- A new "9X" version will eventually exist
  - Many Existing Entries will be removed
  - Some new Entries will be required!
- Ada83-based CIFO is needed now
Future Efforts

- Distributed Systems
- Multiprogramming
- Multi-level Security
- CIFO test suite for "conformance"
- Continuing refinement of existing Entries

Concluding Remarks

- Some controversial interfaces are defined
  - Low level asynchronous task control
- Some questionable interfaces are defined
  - Procedures that require paradigm shift at runtime
- Still the best approach available
  - Some proven, very useful interfaces are standardized
  - Meets the needs of the realtime community now
- Can serve as a bridge for Ada9X
  - Some interfaces are now in Ada9X, in one form or another
  - Education re: issues addressed by CIFO as intro to Ada9X
Other ARTEWG Activities

- Catalog of Runtime Implementation Dependencies
- Framework For Describing Ada Runtime Environments Model Runtime System Interface (MRTSI)
- Ada9X Revision Requests
- Ada9X Realtime facilities design review

Where to Get a Copy

- Send a self-addressed, postage-paid ($3.00) envelope to

  Mike Kamrad
  Paramax Electronic Systems
  M/S U1M30
  PO Box 64525
  St. Paul, MN 55164-0525

  No Checks or Cash!!