Final Minutes of the 2. ARG meeting, Montreux, Switzerland, 15-17 June 1996

ATTENDANCE:

Kiyoshi Ishihata, Vincent Cellier, Pascal Leroy, Stephen Michell, Ted Baker (15 June only), Mike Kamrad, John Barnes, Bob Duff and Erhard Ploedereder.

Guest: Dave Emery (15 June only)

WG9 RESULTS:

Erhard opening the meeting with a review of the results of the ARG motions to WG9 for approval of the AIs. The following AIs are now unanimously approved by WG9:

AI95-00002: Subunits of a preelaborated subprogram
AI95-00026: Stream_IO.Read and Stream_IO.Write advance the current index
AI95-00027: Typo: is access all {Ada.}Streams.Root_Stream_Type'Class
AI95-00029: Does Complex_IO handle extended real literals?
AI95-00036: What are the rules for named notation in pragmas?
AI95-00038: Mapping between Interfaces.C.char and Standard.Character
AI95-00039: Effect of Update(Null_Ptr,...)
AI95-00043: Predefined Operators for Generic Formal Array Types
AI95-00046: Ambiguous "only"; please clarify intent
AI95-00047: {user-defined} Read and Write attributes
AI95-00049: Reserved_128, etc.
AI95-00050: External files of Standard_Input and Standard_Output
AI95-00053: Case sensitivity of Wide_Value and Value attributes
AI95-00056: Create(Mode => Append_File)
AI95-00057: Text_IO.Flush should use IN mode
AI95-00062: Derived access types share the same pool
AI95-00069: One queuing policy per partition.
AI95-00082: The PCS may be defined by the user.
AI95-00083: Aggregates of a controlled type
AI95-00091: Pragma Locking_Policy Cannot Be "In" a Program Unit
AI95-00093: Float_Type'Small
AI95-00101: Abort_Task has a parameter of mode 'in'.

and so are the presentation AIs:

AI95-00005: Incorrect reference in AARM:
AI95-00028: Typo: "pragma" should be in boldface
AI95-00030: The word "prefix" should be in sans-serif font.
AI95-00052: adainit, adafinal should appear in the index
AI95-00066: Incorrect syntax in example -- remove "limited"
AI95-00068: "It also defines [one such policy]{two such policies}."
AI95-00070: Incorrect Picture String Example

AI95-00081: Integer_Text_IO, etc. not listed in A(2)
AI95-00088: Index bug: main subprogram

For AI-62 the approval is subject to the correction of the example in the AI (a typo "Free (X)" should read "Free (Y)").

Compared to the original list of AIs submitted to WG9, AI-1, AI-48 and AI-51 were not brought up for a vote by WG9, due to the existence of "No"-votes in the ARG letter ballot: AI-01 was held back waiting for Bob Eachus and Norm Cohen agreement; Erhard presumes that this won't require another ARG letter ballot. AI-48 is hopelessly deadlocked in the ARG ballot; a quick straw poll at WG9 yielded a similar 3-2-3. AI-51 requires further discussion.

An issue brought by the French delegation against AI-82 and its prohibition of calls on implementation-defined children of System.RPC was briefly discussed at WG9, but the AI was approved regardless. Similarly, AI-27 was briefly discussed prior to approval.

The French delegation showed the following example of a circular definitions and asked for its impact on applying the AI-64 resolution:

```
type PTR is access function return Integer:
function F return Integer;
P: PTR := F'Access;
function F return Integer renames P.all;
```

Since the issue could not be resolved on the spot, the AI was sent back to the ARG without WG9 vote. Pascal points out that this issue is being addressed in AI-135.

As expected, WG9 did not want to discuss AIs relating to the AARM.

After a short discussion, it was tentatively decided that WG9 approved AIs should be moved to a separate directory on the sw-eng repository.

URG, UIS, and AIS IN HTML

Dave Emery requested time at the beginning of the ARG meeting to propose several activities for the ARG to consider:

1. Motion to Incorporate UIs into AIs: Dave Emery has suggested that due to the lack of URG resources and the inclusion of Implementation Advice in the current RM it would appear the UIs have a formal home in the ARG. Rudolf Landwehr had volunteered at WG9 to examine the current list of Ada83 UIs to determine which have already been addressed in the current RM and those which required further action; electronic copy exists on the sw-eng site. Dave Emery recommends that the unaddressed UIs analysis start in the ARG and then seek consensus among vendors for solution.

Discussion: Bob Duff said he remembered one existing AI (AI-131) that applied to Implementation Advice. He said the electronic support system he has can not handle "UI", only "AI". Although there was no formal vote, the consensus at the meeting was that Implementation Advice is the place where the topic of UIs could be handled. Mike Kamrad

raised the question of participation and ownership of vendors in the decision making process; Erhard said that he has tried to activate the ARA to get more vendors involvement in the ARG, but has been singularly unsuccessful to get any action. No real issues were raised in objection to accepting UIs by the group, since there is no chance that a URG will be formed. The group tentatively agreed to deal with UIs as AIs in Implementation Advice category; it was also suggested that some notation be added that indicates compiler (non-)uniformity on a UI issue.

2. AIs on the WEB: Dave Emery makes the case for the publication of AIs/UIs as Web pages. He claims that he has software that is nearly ready to convert the UIs/AIs format into HTML format in fairly automated fashion. Dave expects the AdaIC to do the work to put the AIs onto the net. (This was discussed at WG9 as well, resulting in this suggestion and an action item for Erhard to pursue this idea with the AdaIC.) Erhard then disclosed that he has been approached by a commercial effort to do a Nyberg-like electronic version of the updated RM which would provide the RM as HTML pages. All members of the ARG would have access to the Web pages and other stuff for free.

Discussion: Who will maintain the conversion software? Dave made the commitment to build and maintain the software (in Perl now and Ada95 later?). The ARG certainly was unwilling to maintain HTML pages for current and future AIs. If a commercial venture will do it, why should we depend on volunteer service? Bob Duff noted that it would be nice to have an index that shows the AIs by RM section number, as was available for Ada83 and could be easily available from appropriate HTML pages. Only WG9 approved AIs should be made widely available on the Web. This makes such a service rather uninteresting to the ARG.

STANDARD CORRIGENDA

Erhard passed on some news about the process of Corrigenda as briefly discussed at WG9. SC22 has complained that the previous ARG never published anything officially in form of Corrigenda; Erhard said that we must contemplate the periodic publication of Corrigenda. Since there is very little experience in publishing Corrigenda, we should be able to pick the format that is easiest to produce, i,e., more or less current AI format. Erhard will get editorial instructions from Bob Mathis.

REVIEW OF THE AIS

The AIs were reviewed in the following order:

On 15 June - AI-92, AI-121, AI-118, AI-126, AI-41, AI-73, AI-136, AI-74, Pragma Annotate, AI-85, AI-07, AI-63, AI-87, AI-112, AI-132, AI-137, AI-108, AI-64 and AI-135

On Sunday 16 June: AI-14, AI-44, AI-123, AI-65, AI-117, AI-35, AI-97, AI-127, AI-110, AI-126 (revisited), AI-128, AI-89, AI-137 (Revisited), AI-71/72, AI-124, AI-134, AI-104, AI-106, AI-37, AI-139, AI-140, AI-141, AI-40, AI-31, AI-54

On Monday, 17 June: AI-131, AI-132 (revisited), AI-115, AI-03, AI-04, AI-109, AI-107, AI-48, AI-51, AI-133, AI-59

The following AIs on the agenda were not reviewed in the available time: AI-103, AI-125, AI-34, AI-25, AI-86, AI-98, AI-102.

For ease of reference, these minutes report on the discussion of the AIs in numerical rather than chronological order and first provide the following summary of results.

(Note by the Chair: I decided on the classification of AIs with editorial changes into the following two groups after the meeting according to my interpretation of the amount of change. Any voting member has the right to request a letter ballot on any of the AIs in the first group. This includes the "approved as is" AIs.)

The following AIs were approved "as is" or with minimal editorial changes. Unless the editorial review yields a request for letter ballot, these AIs will be regarded as approved by the ARG:

AI95-00003/00 -- Access types declared in shared passive generic packages
    Approved (7-0-1)
AI95-00007/00 -- Enumeration_IO would allow instantiation for an [float] {integer} type
    Approved (9-0-0)
AI95-00014/01 -- ... and {its calling convention} shall not be Intrinsic.
    Approved (8-0-0)
AI95-00031/00 -- Unpacking a record type with primitive subprograms
    Approved (8-0-0)
AI95-00035/00 -- Overriding in Body
    Approved (5-0-3)
AI95-00037/03 -- Can wchar_t be signed?
    Approved (8-0-0)
AI95-00040/01 -- Does <> for a formal subprogram default freeze the actual ?
    Approved (8-0-0)
AI95-00044/01 -- Overriding of Declarations
    Approved (8-0-0)
AI95-00059/00 -- Specifying Storage size for tasks
    Approved (8-0-0)
AI95-00073/00 -- Pragmas are allowed in generic_formal_parts.
    Approved (9-0-0)
AI95-00074/01 -- Pragma Inline Requires an Argument
    Approved (9-0-0)
AI95-00087/00 -- Saving and restoring Current_Output
    Approved (7-0-2)
AI95-00092/04 -- Async Task Control while caller in rendezvous
    Approved (9-0-0)
AI95-00097/03 -- Conversions between access types with different representations.
    Approved (6-1-1)
AI95-00107/01 -- Base attribute for non-scalar subtypes?
    Approved (8-0-0)
AI95-00108/00 -- Inheritance of Stream Attributes for Type Extensions
    Approved (7-0-2)

AI95-00110/00 -- No Constraint Check on 'out' Parameter of an Access Type
Approved (7-0-1)
AI95-00112/00 -- Wide_String file names?
Approved (8-0-1)
AI95-00115/00 -- Controlled types in language-defined generic packages
Approved (7-0-1)
AI95-00118/00 -- Termination signals query of Terminate attribute
Approved (9-0-0)
AI95-00124/00 -- Ligatures Are Allowed in Identifiers
Approved (6-0-2)
AI95-00134/01 -- Source Representation
Approved (7-0-1)
AI95-00137/01 -- Attribute definition clause for Stream Attributes
Approved (6-0-2)
AI95-00139/01 -- Interfaces.C.Strings.Value Raises Constraint_Error when Length is 0
Approved (8-0-0)
AI95-00140/01 -- Semantics of Interfaces.C.Strings.To_Char_Ptr when Nul_Check is False
Approved (7-0-1)

For the following AIs a letter ballot was explicitly requested or the changes are large enough to warrant a letter ballot:

AI95-00004/01 -- Conversions to types derived from remote access types
Approved (7-0-1) letter ballot requested
AI95-00041/02 -- Program unit pragmas in generic units
Approved (9-0-0)
AI95-00063/01 -- Erroneous execution for closing default files
Approved (8-0-1)
AI95-00089/00 -- Float_Random.Value, Discrete_Random.Value
Approved (6-0-2) letter ballot requested
AI95-00104/00 -- Version and Body_Version attributes
Approved (7-0-1) letter ballot requested
AI95-00106/00 -- Freezing Rules
Approved (8-0-0), letter ballot requested
AI95-00109/01 -- Size and Alignment Attributes for Subtypes
Approved (5-0-3), letter ballot requested
AI95-00121/00 -- Pragma Attach_Handler on Nested Objects
Approved (9-0-0)
AI95-00123/00 -- Equality for Composite Types
Approved (8-0-0) letter ballot requested
AI95-00127/00 -- Expected type of a 'Access attribute
Approved (7-0-1)
AI95-00128/00 -- String Packages
Approved (7-0-1)
AI95-00131/00 -- Interface to C -- passing records as parameters
Approved (8-0-0)
AI95-00136/01 -- Placement of Program Unit Pragmas in Generic Packages
Approved (9-0-0)

The following AIs were discussed and either not voted upon or considered particularly difficult to warrant special attention by the ARG:

AI95-00048/03 -- Can an RCI unit be a library subprogram?
deadlocked
AI95-00051 -- Size clauses for objects specifying large sizes
tabled; to be voted jointly with AI-109
AI95-00054/02 -- When is a Small clause allowed?
tabled; action item: Dritz, Taft, Dewar, Myers
AI95-00065/00 -- Implicit /= is a legal dispatching operation
tabled, also see AI-117
AI95-00071/01 -- Correction to the Valid function in COBOL Interface
tabled; old action item for Dewar, Brosgol, Eachus
AI95-00072/01 -- Clarification of result length for conversions in COBOL Interface
tabled; old action item for Dewar, Brosgol, Eachus
AI95-00085/02 -- Questions about Append_File mode
tabled; action item for ??
AI95-00117/00 -- Convention of an overriding dispatching operation
tabled; action item for Duff
AI95-00126/02 -- Remote_Types Packages
tabled; action item for all
AI95-00132/00 -- Exception raised at end of text stream
Rejected (1-4-3)
AI95-00133/01 -- controlling bit ordering
tabled; action item for all
AI95-00141/01 -- Exceptions Raised by Interfaces.C.Pointers
tabled; action item: Leroy

ANNOTATE PRAGMA

Stephen Michell presented the HRG proposal for a pragma Annotate (see attachment). The HRG is pushing for this pragma to be a predefined pragma (disallowing compilers to use the pragma name for other purposes), presumably as an Annex H feature. An alternative approach would be to accept it as an implementation advice, introducing it as a uniformity feature.

Several issues were discussed or uncovered:

1. Issuing an AI declaring the pragma predefined, thus creating a (first) extension to the language. It was noted that this pragma is only useful to the user, if the Ada implementation also supports ASIS- or Diana-like access to the semantically attributed internal representation, and if there are tools that access this information. Many "if"s for a predefined feature. Nevertheless, the principle intent of the pragma had reasonable support within the ARG, but there were serious doubts that the pragma would technically achieve this intent.

2. Technical issues: There was some discussion about the syntax of the pragma, which is a bit unusual, although within the syntactic rules for pragmas in general. A much more serious objection was the fact that annotations given for subprograms along with their specification would not have visibility of the subprogram parameters. It was also observed that the use of

expressions might seriously limit the expressiveness, e.g., it would be most desirable to give annotations about types, exceptions, subprograms, etc., none of which would fit the mold of semantic processing of expressions by a compiler. Some issues of overload resolution involving universal types were raised, as were problems of overload resolution in the absence of an expected type, as the pragma does not provide such expected types. In short the expression syntax appeared too limiting to be useful for the intended purpose of achieving name binding by the compiler on these expressions. Yet, if any string arguments were used, then "the game would be lost", since ASIS or the tool would then have to provide access to/implement the full name binding/overloading algorithm anyhow to correlate the string contents to the program, in which case there would be little benefit in having an expression syntax in the pragma at all. Further, it was observed that, compared to present style annotations, using "--# " and suchlike, the pragma notation is quite inconvenient for users. Some ARG members recommended that the pragma be experimented with in actual implementations before considering standardization via the ARG route.

The proposal to accept an AI for a predefined pragma Annotate was defeated 1-5-3. The proposal to consider such a pragma as a UI/AI for Implementation Advice sometime in the future in a formal AI/UI format was approved 7-0-2.

NEXT MEETING

The next meeting will be either in October in New England or in December in Philadelphia. To reflect the wishes of absent members, a decision is to be coordinated soon by e-mail.

ACTION ITEMS

Pending old Action Items:

Bob Eachus: AI-71, AI-72
Gary Dismukes: AI-33
Robert Dewar: AI-71, AI-72
Ben Brosgol: AI-71, AI-72

New Action Items:

Eachus: AI-89
Dewar: AI-54
[Landwehr: analyze old UIs; post remaining ones to ada-comment]
Leroy: AI-141
Michell: comment on AI-51
Taft: AI-44
[Dritz: AI-44, AI-89]
[Myers: AI-44]
Duff: post editorially revised AIs, create separate directory for WG9-approved AIs
AI-***** :-)
Ploedereder: AI-41, comment on AI-51, minutes, initiate letter ballots, prepare for next meeting, contact AdaIC re HTML AIs, contact non-ARG members about

their action items, get corrigenda format from Bob Mathis

all: AI-126, AI-133, AI-141
editorial review of all approved AIs, as soon as they are posted
letter ballots as soon as initiated

==========

DETAILS OF THE AI REVIEW (by AI order)

AI-03

Discussion was centered on understanding the consequences of this AI; no other alternatives were presented. Approved 7-0-1.

AI-04

Erhard would like to get input from all ARG members on this issue before sending it to WG9. Approved 7-0-1 with some very minor editorial changes and request for letter ballot.

AI-07

Just a typo AI; approved without discussion 9-0-0.

AI-14

Approved 8-0-0 without discussion. Note, however, substantive discussion on AI-117.

AI-31

Approved 8-0-0 with the note that this was so obvious that it didn't require an AI.

AI-35

This AI confirms to implementors that implementation can be done as implied in the RM (and AARM). Approved 5-0-3 subject to a change in title, more appropriately reflecting the content.

AI-37

Approved 8-0-0 with the following editorials changes:

1. Change subject to positively say the both nul and wide_nul are what you expect
2. Add angle brackets around "implementation defined" in the Wording
3. Add note to Wording that the Implementation Advice sentence is in addition to the existing Implementation Advice.

AI-40

Approved 8-0-0 as a Ramification.

AI-41

Erhard said that he would make some editorial changes to reflect input received since the posting of the revised version. Some members encouraged changes that eliminated some of the ambivalent wording in the discussion. There was general agreement on the conclusion of the AI, since it preserved the most functionality and the flexibility that an implementation can still define new pragmas that apply to all instances. Approved 9-0-0 with editorial changes to be submitted.

AI-44

There was some discussion on the issue of the hiding of inherited subprograms by statement identifiers and the general model spelled out for hiding in the RM, which also impacted the "/=" rules. The discussion on whether specific RM wording should be created to reformulate the model as suggested by Tuck ended with the impression that there is no compelling reason to do so, since the AI covers all the necessary points.

(Note: the rules (Item 3 in the enumerated list of rules) for tagged types makes sure that new components of a derived type are covered by "inherited" equality.)

Approved 8-0-0.

AI-48

AI-48 is deadlocked in ballot. There is no new technical information available or likely to be forthcoming to swing votes. A procedural resolution is needed.

It was suggested that the intent of the AI be reversed and see how the vote would go. At the meeting, this reversed AI was voted 3-2-3, again failing to get real consensus. Erhard asked Bob to rewrite the AI to reverse the intent of the AI and submit for letter ballot. If no strong majority is attained in the letter ballot, then a vote at the next ARG meeting between the old and the new version of the AI WILL resolve this issue for submission to WG9.

AI-51

Both Erhard and Stephen (both dissenters) will supply their reasons ASAP for not approving. Both have concerns about the semantics stated for aliased objects. In addition, Erhard wants to tightly connect the approval on this AI with AI-109 (size of subtypes) because they are so interrelated that one should not be approved without the other. There was agreement with this position and no motion was made to vote on the AI at this meeting.

AI-54

Pascal brought up the issue of Ada83 incompatibility implied by this AI with respect to derived fixed point types. The group concurred that further analysis by numerics semantics expert (Dritz, Taft, Dewar, Myers) is needed. The term "model number" (in the Discussion) should read "values", since fixed point types no longer have model numbers. The AI is tabled until the analysis is completed.

AI-59

This is a trivial confirmation. Handling these (like AI-31) is becoming time consuming for both Bob and the ARG. Consequently Bob was requested to create a "trivial conformation" AI to collect all conformations like this one in the future. The AI was approved 8-0-0.

AI-63

Most found the Question portion to be hard to understand, but yet everyone seemed to be satisfied with the AI Summary and Recommendation. Approved 8-0-1 awaiting editorial changes, in particular expansion of the Discussion section.

AI-64 and AI-135

(Note AI-135 was not on the agenda, but its relation to AI-64 sent back to the ARG by WG9 made it relevant to the discussion.)

The group tried to see if AI-135 handles the problem identified by Jean-Pierre Rosen at the WG9 meeting (which essentially turned up a prima-face unpleasant consequence of a definition by equivalence as postulated by AI-64-- a situation that made the AI immediately suspicious, according to Erhard). The ARG looked at freezing rules and at elaboration checks for the completion of a subprogram by renaming-as-body and how they might be used as a mechanism for detecting the circular definition in the example. None were found. There were arguments, pro and con, over whether infinitely recursive calls are the appropriate semantics for circularly defined subprograms and whether there is a possibility to render this situation illegal in general. Ted Baker argues they aren't the same; he also dislikes the probability of the creation of extra call frames generally implied by this AI; Bob Duff counters that the implied "wrapper call" could be easily "inlined" by the compiler, since the renaming can be given in the package specification. Ted argues that compilers should be able to detect the

cycle for statically defined body completion. Bob argues that this pathology shouldn't be handled any differently than the way other cycle detection can be handled.

Since no existing rules were found to make the example illegal, it was agreed that the current AI-64 renders any call to the subprogram into an infinite recursion. No vote was taken, since no real alternative resolution was offered. (The default action will be to bring this AI back to WG9 unchanged.)

AI-65

A discussion of the implementation model for dispatching and, in particular, for the implementation of the "/=" operation through a dispatching call on "=" let to a closer examination of RM 6.3.1(4-10) in relationship to this implementation model. It was then discovered that the problem was not just with 6.3.1(6) and ist obvious conflict with 3.9.2(10), but also with 6.3.1(8), for which no one could find a justifying reason on the spot. The following example was produced later in the meeting:

```
type T1 is ...
proc P (X: T1);
function F return T1;

generic
  type T2 (<>) is new T1 with private;
  A: T2;
package G is

  type T3 is new T2 with ...
  --  F and P are inherited, but are they intrinsic or Ada ?
  procedure P (X: T3);  -- illegal, if above says 'intrinsic' ???
```

There was general agreement with the intent of the AI 65 Summary. It was also obvious that AI-117 was closely related and that the general model of conventions of dispatching operations deserved another integrated examination to fix various inconsistencies. Bob Duff was tasked to expand AI-117 to a more global discussion and resolution of the issues that have been raised. AI-65, which would presumably be subsumed by the revised AI-117, was tabled.

AI-71/72

These AIs were tabled because there was no COBOL expertise in attendance. ARG expects one or more COBOL experts (Dewar, Eachus, Brosgol) to live up to old action items and help confirm the content of these AIs.

AI-73

Approved 9-0-0 without discussion.

AI-74

This is a confirmation of the RM, despite the non-uniformity with other program unit pragmas, like Pure. Approved 9-0-0.

AI-85

Tabled until the persons tasked already at the last meeting provide input on possible resolutions.

AI-87

No discussion; approved 7-0-2.

AI-89

Pascal is convinced that Robert Eachus has a valid point. The subsequent discussion made it clear the AI should be reversed as per A.5.2(45) where documentation is required to state the nature of strings that Value will not accept without raising Constraint_Error. The intent of the revised interpretation is that it is implementation defined under what circumstances "invalid" strings passed to Value raise Constraint_Error. Particularly Eachus and Dritz should take a close look at the AI when amended. Approved with the altered conclusion (6-0-2) and a letter ballot requested.

AI-92

Ted Baker remembers that Tuck tried to interpret the current wording as justification of full transitivity of priority inheritance. Bob Duff remembers that the last meeting had determined a different interpretation of the RM by the ARG. Ted stated that except for bare machine implementations most implementations of the RT Annex will be done on top of an existing OS and will not be able to comply with a strict interpretation. It would be a real shame to deny these implementation legitimacy, due to real operating system restrictions (like those covered by old AI-325). The group decided that the broader issue of interpretation due to OS restrictions should not be decided by this AI. A discussion on whether a separate, more general AI was needed in the style of AI-325, was inconclusive. On AI-92 the ARG agreed that the Summary is correct but the title should be changed to cover the broader issues raised by the e-mail on the AI and to reflect the elimination of transitive task priorities. Approved 9-0-0 with this and a few minor (spelling) editorial changes.

As an aside, the FRT has already approved one exception to the features of the RT Annex (Seven priorities instead of 31.) for one implementation. So apparently sensible leniency in interpreting the features of the RT Annex is already practiced by the validation process.

AI-97

This is a rewrite where the new version specifically deals with conversions between access types of conventions other than Ada. An editorial change will add "other than convention Ada" after "Convention" in the Summary. There was a desire from Stephen Michell and Erhard to have the semantics implementation-defined rather than unspecified; others didn't care. Approved 6-1-1 with the above change to the Summary.

AI-104

The example below was created to enumerate the cases for the first part of the AI:

```
package P is
  package P1 is
    -- no body required
  end P1;
  ...
end P;
```

when
1. P has no body, then P1'Body_Version is different from any value for Body_Version for P or P1 in a version that has such bodies.
2. P has a body and P1 has no body, then P1'Body_Version = P'Body_Version
3. P1 has a body nested in the body of P, then P1'Body_Version = P'Body_Version
4. P has a body and P1 has a separate body, then P1'Body_Version is determined from the separate unit

There was general agreement on the first paragraph of the Summary. An addition to the AI needs to cover the instance when pragma Import acts as a body.

On the second part, Erhard believes that the first statement of E.3(5) prevents smart compilation from being applied, as this example illustrates:

```
with P, Q;
package Ris
  function F(X: P.A) return Integer;
  -- no references to Q!!!
end R;
```

Erhard claims that any change to Q, which results in a new version for Q, should not mandate a new version for R. (Bob Duff stated that new version is synonymous with change in the object code of a program unit but it's not possible to state that in the RM. Erhard says that this is one obviously possible interpretation, but it should neither be the only one, nor a mandatory one.). Bob recalls that this first sentence of E.3 (5) was added for further clarification on the second sentence. It is too general. A solution is to constrain it to those

situations where there is dependence and the dependent compilation unit changes in a "semantically significant" way. An alternative is to strike the first sentence, leaving the liberal interpretation of compilation unit in the second sentence to handle this situation. The latter alternative is recommended.

Pascal recommends that in E.3(5) it should be unspecified instead of implementation-defined for when the version changes, as it may be next to impossible to describe exactly when the version does or does not change in the presence of smart recompilation strategies. There is general agreement on this point.

Approved (7-0-1) "in principle" with the recommended changes to E.3(5) added to the AI, with a letter ballot requested.

AI-106

A number of editorial changes were decided for this AI:
1. !difficulty: high
2. change "things" to something more appropriate.
3. Make the 3. paragraph of the Summary relative to canonical semantics and include implicit calls
4. The 3. paragraph of the Summary belongs (also) under Wording
5. Change various occurrences of "Acc.all" to "X.all"
6. Put parenthesized [yes], [no] answers into the Question section as in "-- illegal [yes] "

Number 6 is generally recommended on all AIs to improve readability.

Approved (8-0-0) "in principle" with editorial changes and the addition of a Wording section. To be confirmed with a letter ballot.

AI-107

Approved 8-0-0 with the following editorials:
1. delete second paragraph of the Question
2. explain the difficulties with 'Base on composite types
3. Mention the T'Base'Size example, admit that it creates an incompatibility, but point out that 'Size is already incompatible.

AI-108

The discussion was largely centered on understanding the write-up, but yielded no disagreement with the AI. Approved 7-0-2 with very minor editorial changes (e.g., "that that" in the Discussion).

AI-109

Bob points out that this AI fixes a inconsistency in the RM between 13.1(14) and 13.3(55). This examples illustrates the problem:

```
type T is range 1..10;
for T'Size use 32;
subtype S is T [range 1 .. 10];
```

What is S'Size?

13.1 (14) makes S'Size = 32. 13.3 (55) makes S'Size = 4.

Furthermore, the following additional declaration was considered:

```
subtype S1 is T range 1..5;  -- S1 is not statically matching T
```

There is agreement that S1'Size = 3.

Erhard observes that these results (in either combination) are not exactly intuitive. There was some uneasiness about the whole model of statically matching subtypes in the context of representation determination, but nobody came forth with a consistent alternative.

The AI takes the interpretation of giving 13.1(14) precedence. First, for aliased objects, statically matching subtypes at the logical level need to also mean matching sizes (Aliasing depends on statically matching subtypes). Second, if a subtype has a different size then its parent type, it would produce a surprising contradiction to the apparent logical statically matching subtypes, i.e., equal Size should not be an additional criterion for matching subtypes, but rather a consequence. The chosen interpretation will prevent this surprise. The last paragraph of the Summary supports this interpretation.

Now on to related issues, the first being packed arrays or records: Paragraph 2 of the Summary is intended to confirm that the size of records and arrays for packing purposes should follow the rules as stated in 13.2 (7-9), namely the size for packing purposes is the minimum size of the array or record when the length is less than a word size; otherwise the implementation may round the size up to the smallest number of words that can contain the array or record.

The third paragraph confirms that users may apply Size to the types, that obey the rules found in 13.9 (17), to at least confirm the default size that the implementation would choose anyhow. This enables the user to make an assertion about the type size.

It was decided that the third paragraph come under the heading of Implementation Advice, not Recommended Level of Support.

Erhard called for individual votes on the intents of paragraph 2-4 of the Summary. Paragraph 2 was approved 6-0-2; paragraph 3 was approved 8-0-0; paragraph 4 was approved 5-0-3. The AI as a whole with various editorial changes was approved 5-0-3; a letter ballot is requested.

AI-110

Approved 7-0-1 with an editorial change to the example in the question section. The change will show an assignment of a 10 character array in the procedure P.

AI-112

Approved without discussion (except for correction of a typo) 8-0-1.

AI-115

Erhard is concerned about the sweeping nature of the Summary (are we really sure that there are no generic packages with types that should reasonably be implemented by controlled types ?) but all others were reasonably convinced that the enumeration in the first Keith Thompson comment gets all the relevant instances. Bob Duff will double-check that enumeration. Also more RM references (such as A(0) and A.3) will be made to make the AI evident to the users and implementors. A note should be added to say that AARM A.5.2(46.a) is utterly wrong.

Approved 7-0-1 with editorial changes.

AI-117

At first, this AI was regarded only as a "convenience" AI. But it is not strictly a key stroke saving AI due to the hierarchy of types which might be based on the same convention. The implicit "inheritance" helps to switch the implementation of the type hierarchy between different programming languages, like Ada and C++. How this interacts with the Intrinsic convention or how it interacts with the example is not well understood. This AI was linked to AI-65 and both were tabled. See the discussion of AI-65 for details.

AI-118

Most of the discussion centered on the ordering of the two sentences of the Summary and how additional wording in the first sentence muddles the meaning of the sentence.

"An action A1 signals an action A2 if A1 is the termination of a task T, and A2 is the expression T'Terminated, and the value of that expression is True."

The last phrase is unnecessary (and redundant). A2 is the evaluation of the expression not just the expression.

Summary of the changes are:

the second sentence should read: "A task T2 can rely on values of variables that are updated by another task T1, if task T2 first verifies that T1'Terminated is True." and it should be the Summary for the AI.

The Wording should add the following bullet, derived from the old first sentence:

An action A1 signals an action A2 if A1 is the termination of a task T, and A2 is the evaluation of the expression T'Terminated.

Approved 9-0-0.

AI-121

Ted lamented that this issue was one that changed often during the language design and that the last position probably was one that left things in an inconsistent state. Originally there was no static attachment until people wanted the ability to preelaborate the attachment of interrupt handlers.

Ted summarized the issue in this way: There are two independent tasks T1 and T2. Each has a protected procedure that they both want to attach to the same interrupt. Assume T1 attaches its protected procedure first. It saves the current interrupt, which is the default interrupt handler. Later T2 attaches its protected procedure and it saves the current interrupt, which is T1's protected procedure. Next T1 leaves the scope where its protected procedure is declared; it restores what it thinks is the previous interrupt handler, namely the default handler (thus violating the LIFO order assumed by the RM model of restauration). This will result in calamity when T2 leaves the scope where its protected procedure is declared. It restores what it thinks is the previous interrupt handler, T1's protected procedure, whose context no longer exists !!

It is clear that Offer's proposal to limit static attachment to only library level POs defeats the purpose of the Attach_Handler pragma; namely localization and temporary "overriding" of interrupt handlers. The implementation is not required to protect the user for bad system programming in which multiple tasks handle common interrupts in an unsynchronized manner.

Discussion of the semantics of the "previous" interrupt handler for finalization of the current handler wandered around several of the proposed alternatives (from the Response section, first the "another possibility", then the "fourth possibility" were examined). It is erroneous to perform the restoring in anything but a strict LIFO order (The AI incorrectly refers to LIFO as FIFO). The discussion moved on to whether the implementation should be required to raise an exception for a LIFO violation and when it should be detected (when T1 terminates or T2 terminates).

Erhard proposes a global stacking mechanism, in which restauration is done in LIFO order for each interrupt, skipping protected operations no longer alive, and ignoring the "previous" notion in the RM. This model provides a well-defined behaviour but it covers up bad programming practice which is not the proper disclosure.

The Summary will say the expected ordering of changing and restoring of protected objects as interrupt handlers is LIFO and anything else is erroneous. "Previous" interrupt handler has the obvious meaning in proper LIFO order. Extraordinary means could be applied to make this case safer but it was decided not to impose this on implementations. Bob will rewrite.

Approved 9-0-0 with letter ballot requested.

AI-123

It appears that the purpose of this AI is to reassure the user that these predefined types are implemented in the right way for composite comparison. This seems to be the exception to the Ada83 upward compatibility rule regarding "composability" of equality of composite types. Namely, in a user-defined composite type, the components of untagged types will use the predefined equality operator unless the user defines an equality operation for the composite type that explicitly invokes the user-defined operators for those untagged components. The user-defined operations for components of tagged types will always be invoked instead of the predefined operations. There are relevant rules for generics as well: If the formal is a generic private (untagged) type and the actual is tagged, the user-defined "=" will be used in the instance, whereas if the actual is untagged, "=" will revert to the predefined "=". Note that an explicit equality operator of a private type is composable if the type uses the predefined equality operation or the type is tagged type. All agreed that Bob should add a definition of composability to this AI.

The five predefined types in the AI Summary were chosen because they are the only ones with explicitly defined equality operator. Its composability behavior needs to be portable across implementation. We reassured ourselves that sensible implementations of these predefined types will provide composable implementations of equality. Either they will be tagged types or use predefined equality, both of which are composable. We requested that Bob put this analysis of these types (including System.Address) in the response. This AI is a binding interpretation instead of a confirmation. Approved 8-0-0. Pending the editorial changes a letter ballot will be taken.

]AI-124

There was spirited agreement. Approved 6-0-2.

AI-126

The discussion focused on whether certain individual library units should be identified as Remote Types packages, especially those for real-time, systems programming applications:

Ada.Real-Time
Ada.Calendar
Ada.Task_Identification

Most of the discussion focused on the partiton-dependent meaning of the values of these package types, as typified by the Task_id type. It appears that values of these types and operations on them have meanings only within the partition in which the packages are included. Consequently it makes sense to exclude them as parameter types in RCI interfaces, since invocation of these operations in another partition is ill-defined.

On the down side, these packages also include types that could safely be used as parameter types of RCI interfaces. It seems a pity that these types are now excluded from acting as remote types. It was observed that the remote types property may have made more sense when applied on a per-type rather than a on per-package basis.

Discussion turned to the analysis of Task_Identification as a Pure or Shared Passive package. It was determined it can not be either because Task_Ids could be expected to be implemented as pointers. The discussion points out that analysis of predefined packages for qualification as Remote Types or Shared Passive was missing in the language definition. This AI is the first attempt to do that analysis. The package Task_Identification should be removed from the Summary list of packages that can be classified by pragma Remote_Types.

Next the Time type from Ada.Real_Time and Ada.Calendar was discussed. The consideration of the Epoch for these packages was one point, as different partitions may have different epochs and hence difficulties to interpret the respective values meaningfully. Operations on these types, such as adding durations, was the other point. It became clear the operations on these types are mathematically meaningful but not meaningful in the context of the usage of time.

Ted pointed out that the _Random packages cannot be remote type packages, since the parameters are supposed to have reference semantics (i.e., generators should never be duplicated).

Initially, Erhard expects that this AI will accumulate the decision making results of this analysis. But on closer analysis we should be resolving the selection of some predefined packages, like the ones in the Summary. Tabled due to exhaustion, to await conclusion later in the meeting.

AI-126 (revisited)

Erhard will request all members of the ARG to examine/analyse the predefined packages for classification as Remote Type or Shared Passive packages.

AI-127

The proposed wording change to 3.10.2(27) needs to be more clearly spelled out or specifically connected to the change in 3.10.2(24).

There is a dual situation for allocators that has not been handled with the wording changes. It appears that changes should be made to 3.9.2(7) and 4.8(3) in the fashion of changes to those proposed for 3.10.2(27) et al.

We feel we have caught all the missing changes (due to allocation). Approved 7-0-1 subject to completing the wording changes and a letter ballot.

AI-128

Several editorial changes were approved:

1. Add an abbreviated list of changes from the Response section to the Summary;
2. Add short answers to each of the questions;
3. Expand the write-up of the fourth change in the Response section to cover other similar operations per comment 96-5490.a
4. fix typos (e.g., "callwith")

Approved 7-0-1.

AI-131

This AI provides Implementation Advice that users can depend on to write their code, indirectly advising the user on style. (Note that users would be wise to consult other Implementation Advice sections to see if they also affect programming style.) Editorials changes are to the title (applying to "in" parameters) and clarifying/extending the Summary. Approved 8-0-0 with editorial changes. Note that this AI fits our informal definition of a UI.

AI-132

The discussion first centered on whether End_Error is an appropriate exception ever to be raised in the context of streams. There appeared to be support for this idea. The second question was whether at such "end" of the stream, End_Error vs. Data_Error should be used to distinguish between "truly nothing there" and "something there, but not enough". An objection was raised against the latter model, since it would require that a 'Read for a composite type would need to remap an End_Error received from any but the 'Read of the first component to a Data_Error. This seemed like an unnecessary implementation burden for relatively little user benefit. It was noted that the issue is only how the default implementation treats end of file; user-defined 'Reads are free to treat end of file any which way they like; it was also noted that this created interesting ramifications for the above model of exception remapping. (Note: For user implementations, fault free input will probably require some buffering of I/O to correctly interpret the condition.) This discussion wandered throughout the RM among 13.13.1 (6), 13.13.2 (35), A.13 (13, 16, 17) in trying to understand the RM intent and its consequences without convergence towards a single answer.

AI-132 (revisited)

Erhard would like to get closure on this AI, since otherwise it will require assignment for further study and rewrite. It was pointed out that Text_IO supports End_Error over Data_Error and there is an Ada83 AI (AI-037) that reenforces this convention. In a straw vote, the current AI was rejected 1-4-3. Since the discussion did not converge, Bob volunteered to rewrite this AI with all alternatives, such as Data_Error and End_Error (must or may) and so on. Tabled until the rewrite is completed.

AI-133

Tabled until members of ARG can analyse and submit comments on this topic.

AI-134

Approved 7-0-1 without discussion.

AI-136

The ARG consensus after some discussion supports the Summary of this AI. However, the ARG would like to see content in the Wording section. The intent is also to insure that the pragma is in the visible part and not the private part. Also the wording of the Summary should replace "is supposed to" by "must". The new Wording will await editorial review. Approved 9-0-0 on the intent.

AI-137

While there was no disagreement on the content of the Summary, the discussion migrated to the larger issue of how derived types whose parent type has user-defined primitive operations can deal with the representation issues. A vote was delayed to the next day.

AI-137 (Revisited)

Approved 6-0-2 without further discussion.

AI-139

Since the subject of this AI is never explicitly stated in the RM, the category should be changed to Ramification. Approved 8-0-0 with the change to Ramification.

AI-140

Approved 7-0-1 without significant discussion.

AI-141

The discussion uncovered a larger issue of style and bad architecture. The style issue concerns the declaration of exceptions in a generic package and thus the creation of distinct exceptions for each instantiation and the ensuing explosion of handler choices. The bad architecture issue is that a user of the Pointer package must import the otherwise unrelated Strings package for the sole purpose of gaining visibility to the exception Dereference_Error, raised by interfaces in the Pointer (!) package. This is not just very poor interface design, but it also has the potential of carrying a penalty in object code size of linked programs. Making changes now would cause the least amount of problems and predefined packages should exhibit decent packaging.

Bob argues strongly that nothing is (sufficiently) broken and that the practical use will not be causing real programming problems. The proliferation of exceptions will not cause problems because they will all be handled globally by "others" choices. The proposed change amounts to incompatible, gratuitous change.

The regrouping of exceptions was approved by 5-1-2 with the proviso that such a change should be discussed by the entire ARG, preferably by e-mail before the next meeting.

If the regrouping/moving of exceptions is approved, then the group approved 7-0-1 moving all four of them to Interfaces.C to match similar style in other predefined package structures.

This AI is tabled for the rewrite of the AI (by Pascal Leroy) based on moving the exceptions.