Ada 9X Project Report

Ada 9X Requirements Rationale

May 1991

Office of the Under Secretary of Defense for Acquisition

Washington, D.C. 20301

Approved for public release; distribution is unlimited.

Ada 9X Project Report

Ada 9X Requirements Rationale

May 1991

Office of the Under Secretary of Defense for Acquisition

Washington, D.C. 20301

Approved for public release; distribution is unlimited.

Copyright © 1991 by Carnegie Mellon University.
Reprinting permitted if accompanied by this statement.

May 1991

Ada 9X Requirements Rationale

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Copyright © 1991 by Carnegie Mellon University.
Reprinting permitted if accompanied by this statement.

Table of Contents

1 Introduction 1

2 General Requirements 3
2.1 Presentation Requirements 3
2.2 Efficiency, Simplicity, and Consistency 5
2.2.1 Direct Declaration of Generic Subprogram Bodies 8
2.2.2 Default Initialization for Any Non-Limited Type 8
2.2.3 Restrictions on Subprogram Parameters 9
2.2.4 Extend Use of Representation Attributes 9
2.2.5 Allow Use of Private Type Before Its Full Declaration 9
2.2.6 Allow Components with the Same Name 9
2.2.7 Allow Accept Statements in Nested Subprograms 10
2.2.8 Using Real Literals with Fixed Point Operations 10
2.2.9 Controlling the Effect of Pragma INLINE 10
2.2.10 Allow Type Conversions in More Contexts 10
2.2.11 Private Task Entries 11
2.2.12 Loops over Non-Discrete Types 11
2.2.13 Permit Other Bracketing Characters 11
2.2.14 Internal Coding of Enumeration Values 12
2.3 Error Detection 12
2.4 Controlling Implementation-Dependent Choices 13

3 Requirements for International Users 17
3.1 International Character Sets 17

4 Support for Programming Paradigms 19
4.1 Subprograms 19
4.2 Storage Management 20
4.3 Composition of Program Units 22
4.4 Generics 25
4.5 Exceptions 27
4.6 Input/Output 28

5 Real-Time Requirements 31
5.1 Time 31
5.2 Task Scheduling 32
5.3 Asynchronous Control of Execution 34
5.4 Asynchronous Communication 34

6 Requirements for System Programming 35
6.1 Unsigned Integer Operations 35
6.2 Data Interoperability 35

Table of Contents

6.3 Interrupts 36
6.4 Dynamic References to Global Objects 37

7 Requirements for Parallel Processing 39
7.1 Shared Memory 39
7.2 Massively Parallel Architectures 39
7.3 Vector Architectures 39
7.4 Configuration of Parallel Programs 40

8 Requirements for Distributed Processing 41
8.1 Distribution of Ada Applications 41
8.2 Dynamic Reconfiguration of Distributed Systems 41

9 Requirements for Safety-Critical and Trusted Applications 43
9.1 Predictability of Execution 43
9.2 Certifiability 43
9.3 Enforcement of Safety-Critical Programming Practices 43

10 Requirements for Information Systems 45
10.1 Handling Currency Quantities for Information Systems 45
10.2 Compatibility with Other Character Sets 45
10.3 Interfacing with Data Base Systems 45
10.4 Common Functions 45

11 Requirements for Scientific and Mathematical Applications 47
11.1 Floating Point 47
11.2 Representation of Arrays 48

12 Efficiency, Simplicity, and Consistency Issues 49
12.1 Efficiency of Executed Code 49
12.1.1 Access to a Task Outside its Master (see RD A.1.1) 49
12.1.2 Null Ranges (see RD A.1.2) 49
12.2 Understandability 49
12.2.1 Elaboration Order (see RD A.2.1) 49
12.2.2 Later Declarative Items (see RD A.2.2) 50
12.2.3 Visibility of Literals and Operations (see RD A.2.3) 50
12.2.4 Obsolete Optional Bodies (see RD A.2.4) 51
12.2.5 OTHERS Choice in Aggregates (see RD A.2.5) 51
12.3 Generality 51
12.3.1 IMAGE and VALUE for Real Types (see RD A.3.1) 51
12.3.2 Exception Handlers in Accept Statements (see RD A.3.2) 51
12.3.3 RANGE Attribute for Scalar Types (see RD A.3.3) 51
12.3.4 Permit ‘‘raise ... when <condition>’’ (see RD A.3.4) 52
12.3.5 STORAGE_SIZE for Task Objects (see RD A.3.5) 52

12.3.6 Explicit Type Conversions in Static Expressions (see RD A.3.6) 52
12.3.7 Use of a Subprogram Name in its Specification (see RD A.3.7) 52
12.3.8 Default Names for Generic Formal Parameters (see RD A.3.8) 53
12.3.9 Ability to Redefine ‘‘=’’ (see RD A.3.9) 53
12.3.10 Reading OUT Parameters (see RD A.3.10) 53
12.3.11 Implicit Subtype Conversions (see RD A.3.11) 53
12.3.12 Negative Literals in Loops (see RD A.3.12) 53
12.3.13 Naming Syntactic Items (see RD A.3.13) 54
12.4 Usability of Ada 54
12.4.1 Completion of Subprogram Declarations (see RD A.4.1) 54
12.4.2 Completing Incomplete and Private Types by Subtype Declarations (see RD A.4.2) 54

13 Requests Rejected 55
13.1 Requests Rejected: Various Reasons 55
13.1.1 Requests Rejected: Machine Code 57
13.1.2 Requests Rejected: Exit from a Block 58
13.2 Requests Rejected: An Opposing Requirement was Imposed 58
13.3 Requests Rejected: Insufficient Information in the Request 58
13.4 Requests Rejected: Insufficient User Benefit 59
13.4.1 User-Defined Attributes 61
13.4.2 Multidimensional Slices 61
13.4.3 Permit Functions to have Parameter Modes IN OUT and OUT 62
13.4.4 Anonymous Arrays 62
13.5 Requests Rejected: Too Much Implementor Change for the Payoff 62
13.5.1 User-Defined Operators 62
13.5.2 Non-static Objects as Case Labels 63
13.5.3 Discontiguous Subtypes 63
13.5.4 Overload Names of Generic Units 63
13.6 Requests Rejected: Not Sufficiently Compatible with Ada 83 63
13.7 Requests Rejected: Not a Language Issue 64

References 67

Appendix A: Numerical Listing of RRs 68

Appendix B: Numerical Listing of AIs 99

Appendix C: Included Documents 102
C.1 Why Static Expressions Can’t Contain Explicit Conversions 102
C.2 Restriction on Negative Literals 103

Appendix D: KWIC Listing of RR and AI Titles 105

1 Introduction

The Ada 9X Revision process (described in [2]) included the solicitation of *Revision Requests* (RRs) from the world-wide Ada community. Well over 700 RRs were submitted in response to this solicitation, and these RRs formed an important input in determining the requirements to be met by the Mapping/Revision Team. In addition, suggestions for improvement have been gathered over the years in comments submitted in accordance with the directions in the Postscript to the standard [4]. These comments have been grouped into draft Ada commentaries (AIs) in the *study* class. Both the revision requests and commentaries in the study class served as inputs to the Requirements Team.

This document serves several purposes:

- It shows how each RR and study AI is related to the requirements given in the *Ada 9X Requirements* document [9]. The association is given in three ways: in order by requirement number, in order by RR and AI number, and in order by keywords appearing in the title of an AI or RR.
- This document also discusses some alternatives that were considered during the requirements definition phase and in some cases amplifies the discussion presented in the requirements document itself. In developing these comments on the requirements, we have been helped by comments on the requirements document that were submitted by members of the canvass group established for voting on the proposed revision to the standard.

It should not be surprising that it was judged infeasible to meet all requests submitted. Indeed, doing so was impossible when RRs made contradictory requests. Consider, for example, RR-0098, ‘‘Generalize incomplete typing for types other than access or private’’ and RR-0259, ‘‘Incomplete type declarations are dangerous and unnecessary’’.

Some RRs contained multiple requests; for these, each request is listed separately with a letter after the RR number as if the RR were in fact multiple RRs. After eliminating duplicates (6 RRs and 14 AIs), there were 822 RRs and 60 AIs to be considered of which 214 (25%) were rejected. The rest were addressed at least in part by some requirement.

This document is intended to be read in parallel with the Requirements Document. Chapters 2 through 11 of this document correspond to the same-numbered chapters of the Requirements Document; Chapter 12 corresponds to Appendix A of the Requirements Document. Each of these chapters contains the same section titles as the corresponding chapter of the Requirements Document and lists (by title only) the User Needs, Requirements, and Study Topics. Included for each Requirement and Study Topic is a list of all RRs and AIs whose request is met, at least in part, by that item.

Chapter 13 lists those RRs and AIs that have been rejected, providing at least a phrase and in some cases a few sentences of explanation for the rejection. When explanations apply equally to several RRs, such RRs are collected into subsections of that chapter so that the explanation need be provided only once.

Finally, Appendix A lists, in numeric order, all Revision Requests submitted; Appendix B similarly lists AIs. For each item a title is given along with a brief explanation of its disposition:

- For those RRs and AIs whose request was met (at least in part), there is a reference to the Requirements Document that points to the place in that document (a Requirement or a Study Topic) where the topic of the RR or AI is dealt with. Such a reference indicates that the request was met to at least some extent, even if it was not met in its entirety.
- Some requests, usually requests for no change in a specific area, are marked as having been met, with no reference.
- Some requests are marked as rejected, with a brief explanation and a reference to a section of Chapter 13 of this document.

The Requirements Document is referenced either by section number or by citing a specific Requirement or Study Topic. All Requirements and Study Topics are listed in the Table of Contents of [9].

The titles given for RRs and AIs were generated to reflect our understanding of the intent of the request. When an RR covered multiple topics, separate titles were devised for each topic.

Acknowledgments

The initial phase of the requirements process was performed by a team based at the Institute for Defense Analyses (IDA). This team consisted of Cy Ardoin (IDA), Paul Baker (CTA), Douglas Dunlop (Intermetrics), Audrey Hook (IDA), Joseph Linn (IDA), Catherine McDonald (IDA), Reginald Meeson, Jr. (IDA), Steven Michell (Prior Data Sciences: Canada), and Karl Nyberg (Grebyn Corp). The initial team collected the revision requests, classified them, and produced version 1.0 of the requirements in June 1990. This version was not publicly released but consisted of a high-level statement of key requirements distilled from the revision requests, workshops, and various meetings with the Distinguished Reviewers. The IDA version of the requirements was extremely helpful to us in developing Version 3.3 of the requirements (the so-called Working Draft released in September 1990). Without the IDA version as a baseline, it would have been very difficult for us to have produced a satisfactory version by September 1990.

The Working Draft version of the Requirements Document requested comments from the public on a very short time scale. Approximately 1500 comments were received, and just about all of them were constructive and helpful, especially those that suggested revised wording. Anyone comparing the September release with the final release will see many important changes, and many of these changes directly reflect the comments that we received. Although we cannot take the space to list each comment, most commenters will see that their contribution did indeed affect the final version of the requirements. Please accept our thanks and appreciation.

In addition, we particularly wish to thank Norman Cohen for providing many helpful comments on the Working Draft and a draft version of Chapter 9 on requirements for trusted and safety-critical systems.

The Requirements Team consisted of John B. Goodenough (Software Engineering Institute), leader and principal author of the Requirements document and of the Rationale; Arthur Evans, Jr. (Software Engineering Institute), Robert B. K. Dewar (New York University), and Benjamin Brosgol (Alsys, Inc.). We are grateful to Gary Dismukes for providing careful comments on a draft of the Rationale.

2 General Requirements

2.1 Presentation Requirements

User Need U2.1-A: Improve Wording to Reflect the Intended Meaning

Requirement R2.1-A(1) — Incorporate Approved Commentaries

Several revision requests mention problems that are addressed by approved Ada commentaries. Hence, these requests (listed below) are satisfied by the requirement to incorporate approved commentaries into the revised standard.

RR-0013 Allow task activation to occur at a higher priority than task execution
This request appears to be met by AI-00288, which requires that a task's activation occur at either its activator's priority or at its normal priority, whichever is higher.

RR-0023 Require TERMINATE alternative to terminate library tasks
This is already addressed by AI-00399.

RR-0215 Clarify termination of tasks dependent on library packages
This problem is addressed by AI-00399.

RR-0257 Ensure that BOOLEAN and BYTE arrays can be tightly packed
AI-00555, which has been approved by the Ada Rapporteur Group, specifies that arrays of boolean components must be packed with no gaps. AI-00556 addresses the problem of arrays of bytes, but has not yet been approved.

RR-0370C Library level tasks can't terminate
AI-00399 explains when such tasks can terminate.

RR-0496 Clarify termination of tasks whose masters are library units
AI-00399 defines the effect of termination on library-level tasks.

RR-0571B Clarify the effect when the choice in an aggregate is outside the range of the applicable index constraint
AI-00309 deals with this problem.

RR-0581B Clarify the effect of applying pragma ELABORATE to a package that has no body
AI-00236 specifies the effect of the pragma in these cases.

RR-0583 Delete NUMERIC_ERROR if now subsumed under CONSTRAINT_ERROR
AI-00387 recommends that NUMERIC_ERROR be replaced with CONSTRAINT_ERROR.

RR-0724 Need clearer/simpler overload resolution rules, especially for implicit conversion
The problem mentioned here is addressed by AI-00136 and AI-00606.

RR-0769 Correct wording in the definition of ancestor unit
See AI-00482.

Requirement R2.1-A(2) — Review Other Presentation Suggestions

The following RRs mention areas for possible improvement in the wording of the standard. These requests do not generally ask for a change in functionality — only clearer wording is requested. When the Mapping/Revision Team begins to revise the wording of the standard, these requests should be given consideration, although not all of these suggestions should necessarily be followed. For example, several of the requests complain that it is too hard to determine the consequences of language rules. In essence, these requests want the Standard to be written in a more tutorial manner. We list these requests here because they indicate areas in which the wording of the Standard might be made clearer. However, we did not specify that the Standard be written in a more tutorial manner because such a presentation style is inappropriate for a Standard. For example, one goal in writing a Standard is to avoid redundant or alternative phrasings

of rules lest the alternative statements be interpreted differently. While such alternative phrasings may be helpful to readers who are unfamiliar with a language, they are inconsistent with the goal of minimizing possibilities for divergent interpretations of rules. Moreover, being more tutorial would make the Standard longer without making it sufficiently tutorial for many users.

(Note: underlined Revision Requests and AIs contain examples or discussion that may be especially helpful to the Mapping/Revision Team.)

RR-0067 Clarify/define technical terms used
This RR provides some detailed comments that may be useful.

RR-0204 Clarify which fixed point operators are predefined
This RR proposes an improvement to the Standard's Appendix C.

RR-0206 Paragraph numbers should be included in the cross references

RR-0260 The Standard is unclear in various ways
The RR contains several useful suggestions. However, the submitter wants the Standard to be more tutorial, which is probably not possible in a document intended to serve as the specification for a language.

RR-0267 The Standard is confusing in distinguishing specifications and declarations
The submitter wants the Standard to be more tutorial.

RR-0274 The visibility rules could be explained more clearly
The submitter wants the Standard to be more tutorial.

RR-0281 Confusing treatment of term ''delay statement''

RR-0292 Section 13.6 of the standard has no semantic content
The RR notes correctly that the section is, in essence, just a note and perhaps should be so titled.

RR-0298 Clarify classes of objects usable as attribute prefixes
The submitter wants the Standard to be more tutorial.

RR-0301 The wording concerning checking for consistency between compilations can be improved
The RR suggests a helpful rewording.

RR-0305 Clarify wording of FOR loop completion

RR-0309 Ensure all cross references are complete and correct

RR-0350 Clarify wording dealing with default initial values

RR-0436 Clarify task synchronization point inconsistencies
This problem should be addressed.

RR-0500 More terms should be hyphenated to improve clarity

RR-0501 The Standard should be consistent in delimiting section headings

RR-0502 The Standard should be consistent in its use of upper and lower cases

RR-0528 Change Ada character names to recognized names for verbal communication
The problem addressed in the RR is the names assigned to characters in Section 2.1(15) and 2.2(10). The RR cites federal law requiring facilities for the handicapped.

RR-0622 The Standard should use ''metatype'' in describing generic formal types

RR-0757 Clean up definitions of program unit and compilation unit
This clarification may be worthwhile.

RR-0758 Bad paragraph numbering

User Need U2.1-B: Maintain Continuity with the Existing Standard

Requirement R2.1-B(1) — Maintain Format of Existing Standard

During the development of the Ada 9X requirements, there were several intense discussions about whether a formal definition should be provided for the 9X revision. A group at the Destin Workshop [1] recommended strongly that a formal static semantic definition be included in the

revised standard. It was argued that such a definition would reduce the number of ambiguities that would otherwise be present. The Requirements Team and the Distinguished Reviewers decided however, that the development of a formal static semantic definition should not be required as part of the revision process. There were several reasons. First, if the current standard's specification of legal programs was replaced with a more formal notation, this mere change in presentation would give the appearance of great change to Ada 83 whether or not this was in fact the case. Second, restricting the use of formal notation to just a description of Ada's static semantics would not in practice be of significant help to Ada programmers or implementors even though some people have suggested that the large number of official Commentaries [3] on the language indicates that the current presentation method is inadequate or unsuitable. A careful study of these commentaries shows that most of them deal with minor technical corrections where the intent was clear but the wording was imprecise. A formal definition would not necessarily reduce the need for technical corrections.

Other suggestions for reorganizing the standard or for rewriting it in a different style were made by various sources and were rejected for essentially the same reasons. Although there are a number of Revision Requests that, in essence, complain that the RM is hard to understand, it is far from clear that any alternative style or organization would be both easier to understand and technically precise.

Finally, whether or not one agrees with the above arguments, the Requirements Team felt that the Mapping/Revision Team should concentrate on improving the functionality of the Ada language rather than on revamping the presentation of the standard.

User Need U2.1-C: Derivative Use of the Standard

Requirement R2.1-C(1) — Machine-Readable Version of the Standard

Although ANSI/MIL-STD-1815A is available in ASCII format, it was not distributed with text-formatting codes. This increased costs to those who wished to produce a printed version of the standard while maintaining its typographic style. Two revision requests suggested that a machine-readable version of the revised standard be provided and that this version use a standard formatting (or markup) language. The Standard Generalized Markup Language (SGML) [10] was suggested by both revision requests, but the Requirements Team felt it was best to leave this decision to the Mapping/Revision Team. For example, it might be be reasonable to use TEX since this formatting language is widely available and in the public domain.

RR-0318 Make a machine-readable version of the Standard available (with embedded mark-up)
RR-0481 Make Ada documentation available in SGML format

2.2 Efficiency, Simplicity, and Consistency

It was clear from the beginning of the revision effort that the minimal change to the standard would be to at least incorporate wording changes that reflect approved Ada commentaries. This section of the requirements document represents the next level of minimal change, namely, small changes that reflect better insight into rules of the language that incur unexpected penalties in compilation or execution costs, or that have proven to be confusing to users or unnecessarily restrictive. These are the kinds of ''clean-up'' activities that are a normal part of any language

revision effort. The intention in this part of the requirements document was to redress anomalies that could be fixed by small changes to the language.

User Need U2.2-A: Minimize Compilation and Execution Costs

Requirement R2.2-A(1) — Reduce Deterrents to Efficiency

Appendix Section A.1 of the Requirements Document lists some areas in which unlikely interactions between language features incur excessive compilation or execution costs. Revision requests associated with the suggestions in that appendix are discussed in Section 12.1 on page 49.

Only a partial list of potential areas for revision in this area was given in the Requirements Document. Additional RRs that point out potential areas to be considered are listed below. The lengthy discussion in RR-0705 is particularly relevant.

RR-0122 Permit an implementation to reject some integer types as array indexes

RR-0279 If tasks are not used, the run-time system and compiled code should not include code for tasking support

RR-0401 Mixed-base fixed-point operations cannot be done efficiently because of accuracy requirements

RR-592 duplicates the content of this RR.

RR-0574 Inability to eliminate constraint check for OUT parameters

This RR points out a situation in which a redundant constraint check must be performed both inside and outside a procedure.

RR-0693 Parameter passing rules for scalars makes generic code sharing hard

RR-0705 For better performance, remove restrictions on static expressions

The RR provides a detailed analysis of approved AIs in terms of their possible effect on efficiency of both compilation and execution.

RR-0740 For optimization with respect to inlined subprograms, allow merging of scopes

Several revision requests mention a need to increase an implementation's freedom to optimize programs. Other requests state that section 11.6 of the standard already allows too much freedom. Issues concerned with Ada's optimization rules will be the subject of a forthcoming Ada 9X Report. The RRs listed below are those that indicate ways in which section 11.6 may constrain implementations too greatly. RRs complaining that there is too much optimization freedom are cited in association with Requirement R9.1-A(2).

RR-0387 Relax 11.6 optimization rules to allow compiler to do more optimizing

RR-0683 Section 11.6 of the Standard is unclear about what replacements are allowed

RR-0685 Clarify and loosen 11.6 to allow more optimization

RR-0739 Relax 11.6 canonical order rules to allow more optimization

User Need U2.2-B: Ease of Learning

Requirement R2.2-B(1) — Understandability

Appendix Section A.2 of the Requirements Document lists some language rules that have proven to be confusing or error-prone for users of the language and that are therefore candidates for revision. Revision requests associated with the suggestions in that appendix are discussed in Section 12.2 on page 49.

Only a partial list of potential areas for revision was given in the Requirements Document. Additional areas for consideration are listed below.

RR-0094 Make the multiple declaration rules more complete and consistent

RR-0275 Error-prone and counter-intuitive aspects of RENAMES

RR-0344 Need to simplify/relax the conformance rules

RR-0395 Include formal parameter names in parameter/result-type profile

The RR points out that it is illegal to declare two subprograms with the same parameter and result type profile in the same declarative region even if corresponding formal parameter names are different. This illegality seems inconsistent to programmers since such subprograms can be unambiguously called using named parameter associations, and moreover, such overloadings can occur as a result of USE clauses and generic instantiations (see, e.g., 12.3(22)). The Mapping/Revision Team may wish to consider whether this apparent irregularity should be preserved in Ada 9X. The RR points out that allowing such declarations may cause problems with renaming declarations, since if the subprogram being renamed is overloaded in this way, the overloading cannot be disambiguated based on the formal parameter names of the renamed subprogram.

RR-0565 'SMALL is unsuitably defined; need for representation clauses inappropriate

RR-0600 Allow formal parameter names in parameter/result-type profile

See the comment for RR-0395.

RR-0619 Eliminate three replacement characters, stick to normal ASCII

Simplifying the language by removing these alternatives is not upward compatible, but few programs use these replacement characters.

RR-0631 Make conformance rules consistent

RR-0774F Allow aliased exceptions within the same exception handler

It seems that it would be both useful and harmless to allow both P1.END_ERROR and P2.END_ERROR as exception choices in a single exception handler when both exceptions denote IO_EXCEPTIONS.END_ERROR.

Several requests noted the confusion that arises because a fixed point type need not include the values specified in the type's range constraint.

RR-0191 Fixed point model numbers should include the bounds of the type definition

RR-0252D Fixed point type should include the bounds of the range definition

RR-0566 Fixed point model numbers should include the bounds of the type definition

User Need U2.2-C: Generality

Requirement R2.2-C(1) — Minimize Special-Case Restrictions

Appendix Section A.3 of the Requirements Document lists some language restrictions that seem to be surprising to users of the language. For example, the Requirements Document (and one revision request) note that although an accept statement is similar in structure to a procedure body, an exception handler cannot be written after the sequence of statements of an accept body. Revision requests associated with suggestions contained in Appendix Section A.3 of the Requirements Document are listed in Section 12.3 on page 51.

Appendix Section A.4 lists small functional extensions that might make Ada easier to use. Revision requests associated with these extensions are listed in Section 12.4 on page 54.

Only a partial list of potential improvements in generality and usability was given in the Requirements Document. Additional areas to be considered are given below.

RR-0010 Allow the full declaration of a private type with discriminants to be a derived type

See also RR-0423.

RR-0319 Remove arbitrary language restrictions, improve orthogonality
The RR does not give any specific suggestions, but the general intent of the RR is consistent with the requirement for generality.

RR-0341 Allow discriminant value in record aggregate to be non-static
The RR makes a useful suggestion for removing a restriction.

RR-0381 Records should have composed operations with respect to components

RR-0418 Representation clauses for array types need to be added

RR-0423 Remove discriminant restriction on full declarations of private types
The RR raises some points worthy of consideration. It refers to AI-00037 for a complete discussion of the problem.

RR-0522 Allow non-discrete record discriminants

RR-0567 Allow variable declaration to get constraints from initial value

RR-0568 Allow non-nested variant parts in record types

RR-0577 Allow deferred constant of composite type having a component of an incompletely declared private type

RR-0601 Allow library-level declarations to be defined by RENAMES

RR-0610 Why not allow RENAMES for types and subtypes?

RR-0715 Allow user-defined type conversions and attributes for numeric types
The ability to allow programmers to build user-defined types that have the same attributes and conversion notation as the predefined types is attractive as a generalization of the language's existing capability, but it is unclear whether such changes can be made without introducing anomalies.

AI-00280 Allow pragma OPTIMIZE in package specifications

AI-00404 Use of incomplete private types in generic formal part
This AI illustrates an unintended annoying consequence of the rule restricting the use of an incomplete private type.

AI-00519 Default SMALL should be a power of two times the range
This request reflects the need for fixed point types with maximum accuracy for the specified range. This need is in conflict with the Information Systems need for maximum range with only the specified accuracy.

AI-00812 Attributes SAFE_LARGE and SAFE_SMALL should be static

Additional requests pointing out restrictions that might be eliminated or small functional extensions are listed in separate sections below.

2.2.1 Direct Declaration of Generic Subprogram Bodies

Since subprogram bodies can be declared without first giving a subprogram declaration, it seems inconsistent not to have the same shortcut for generic subprograms.

RR-0426B Allow declaration and body to be combined for generic subprograms

RR-0547 Like non-generic subprograms, allow merge of specification/body for generic subprograms

RR-0604 Like non-generic subprograms, allow merge of specification/body for generic subprograms

AI-00382 Allow generic subprogram bodies

2.2.2 Default Initialization for Any Non-Limited Type

Several requests point out that it seems inconsistent to restrict default initialization just to components of records.

RR-0129 Allow default initialization to be specified for any non-limited type

RR-0161 Allow default initialization for any non-limited type

RR-0230 Allow initialization to be associated with any type definition

RR-0456 Allow initialization to be associated with a type definition

RR-0506 Allow initialization to be associated with a type definition

RR-0595 Allow default initialization for all types

RR-0649 Allow default initialization for all types (not just records)

RR-0677 Allow initialization clauses on scalar type declarations

2.2.3 Restrictions on Subprogram Parameters

The following requests note possibly unnecessary restrictions on subprogram parameters.

RR-0103A Allow unchecked conversion for IN OUT and OUT parameters

RR-0239B A renamed type cannot be used in an actual parameter type conversion

RR-0578 Out-mode parameters of limited private types should be allowed
This RR gives a good example to show why this restriction should be relaxed to allow good modular programming practices to be supported.

AI-00473 Any form of actual parameter should be allowed as a default parameter
This AI points out an easily fixed inconsistency in the language.

AI-00840 Allow access OUT parameter as attribute prefix
This AI points out an unneeded and overly restrictive rule.

2.2.4 Extend Use of Representation Attributes

Two requests note that it would be natural and convenient to be able to use representation attributes of composite types in representation clauses.

RR-0048 Extend static expressions to include representation attributes of composite types
Although the request is phrased in terms of allowing generic formal types to be used in static expressions, the example mainly shows a need to ensure that certain representation attributes can be used in static expressions.

AI-00539 Allow use of array/record attributes in representation clauses

2.2.5 Allow Use of Private Type Before Its Full Declaration

Several RRs note that it would be convenient to relax the restrictions on the use of a private type before its full declaration.

RR-0082 Allow declaration of objects of private types in visible package specification

RR-0542 One way or another allow usage of private type before its completion declaration
The need here may be met indirectly by solutions for User Need U4.3-B. AI-00327 contains more detail about the problem.

AI-00327 Instantiating with an incomplete private type
This AI is similar to RR-0542, but contains more detail.

2.2.6 Allow Components with the Same Name

Two revision requests point out why it would be useful to allow different variants of a record to have components with the same name.

RR-0532 Allow same-type record components in different variants to share name
RR-0707 provides a careful analysis of this problem.

RR-0707 Need same-name component identifiers in different variants
This RR gives a careful analysis of the unpleasant workarounds required today, but it is not clear that the implementation impact of this change would be acceptable.

2.2.7 Allow Accept Statements in Nested Subprograms

Several RRs point out that it would be convenient to allow accept statements in subprograms nested within a task, simply because of the beneficial effects on program structure.

RR-0543 Allow accept statements in subprograms nested inside tasks

RR-0580 Allow accepts within subprograms/packages nested inside tasks

AI-00214 Allow accept statements in program units nested in tasks
This AI provides another example in the spirit of RR-0543.

2.2.8 Using Real Literals with Fixed Point Operations

One RR and one study AI pointed out that it was natural to allow the use of real literals with fixed point multiplication and division operators, but the current rules forbid such use.

RR-0591 Allow fixed-point multiply/divide with universal real operands

AI-00262 Real literals with fixed point multiplication and division
This AI makes the same suggestion as RR-0591.

2.2.9 Controlling the Effect of Pragma INLINE

These requests, all concerned with obtaining finer control of the effect of pragma INLINE, are concerned with two issues: 1) inlining some but not all calls to the same subprogram body, and 2) controlling whether a subset of overloaded subprograms are inlined. It is easy to program around the first problem. Mark the first subprogram as being inline, and export a second subprogram, with a similar name, which is not marked as being inline and whose body calls the first. For the second case, it is necessary to be sure the subprograms to be inlined all appear first and that maintenance programmers realize that the order of declarations is important (as is pointed out by RR-0687). The issue is one that deserves attention from the Mapping/Revision Team.

RR-0060 Allow inlining of subprograms from some but not all call sites

RR-0398 Need clearer/more selective rules for pragma INLINE applicability

RR-0575 Need better (more selective) control over inlining

RR-0687 Pragma INLINE should not apply to all overloads; only closest

2.2.10 Allow Type Conversions in More Contexts

RR-0510 points out that in linear algebra subprograms, it would be convenient and often more efficient to be able to re-index arrays directly by subtype conversions when selecting a component, e.g.,

```
VEC_TYPE(VECTOR)(I)
```

Of course, one can define a function whose effect is to apply such a subtype conversion, since

```
FUNC(VECTOR)(I)
```

is quite legal, at least in an expression.[1] If this effect can be achieved indirectly, why shouldn't the language allow it directly and efficiently? Type conversions should at least be usable where equivalent functions are allowed.

[1]The RR points this out very quietly. Lest the significance of the examples given in the RR be missed, we provide a little extra discussion in this section.

One can even argue that type conversions should be allowed on the left side of assignments since subprogram calls can be used to achieve the same effect, e.g., the function

```
procedure ASSIGN_I (TARGET : in out V_TYPE; SOURCE : in V_TYPE) is
begin
   TARGET(I) := SOURCE;
end ASSIGN;
```

combined with the call:

```
ASSIGN_I (V_TYPE(W), ...);
```

has the same effect as:

```
V_TYPE(W)(I) := ...;
```

Since one is allowed to achieve the effect of assigning to a converted array by using a subprogram call, perhaps the language should allow it directly as well.

2.2.11 Private Task Entries

These requests for the ability of a task to have entries that are visible only within the package in which the task is written would extend the expressive power of the language.

RR-0090 Allow some task entries to be visible, some not

RR-0487 Need private task entries for exclusive use within the task

RR-0628 Need private task entries

2.2.12 Loops over Non-Discrete Types

Several RRs request improvements in the loop construct, including the ability to specify a step size and the ability to loop over a list. The Mapping/Revision Team should determine whether these changes are worthwhile, since they were explicitly rejected during the initial design as not being of sufficiently widespread benefit.

RR-0317 Augment Ada's looping: over reals, list items, etc

RR-0615 Define LOOP/UNTIL control structure as in Pascal

RR-0717 Allow specification of a step size in FOR loops

RR-0743 Need to allow increment of something other than one in for loops

RR-0744 Allow for loop to have non-discrete (fixed-point) parameter

2.2.13 Permit Other Bracketing Characters

Several RRs request that programmers be allowed to use parentheses in addition to ''(...)''. This may be a reasonable request to consider given the expansion in Ada's character set (see Requirement R3.1-A(1)). On the other hand, the availability of additional bracketing characters might impair readability by allowing divergent conventions for the use of these characters.

RR-0534 Allow brackets other than ''('', '')'' in aggregates, etc

RR-0556 Parentheses are used for too many purposes in the language

RR-0755 Allow ''['' instead of ''('' for indexed components

2.2.14 Internal Coding of Enumeration Values

Although unchecked conversion can be used to get access to the coding of an enumeration value, several requests ask for the ability to access this coding directly.

RR-0040 Need a way to determine the internal coding of enumeration values
RR-0059 Need an attribute for returning a representation's underlying value
RR-0220 Need way to get the internal code associated with enumeration values
RR-0465 Need a way to get the representation from an enumeration value and vice versa

2.3 Error Detection

User Need U2.3-A: Minimize Consequences of Programmer Error

Study Topic S2.3-A(1) — Improve Early Detection of Errors

Discussions with developers of safety-critical and trusted systems emphasized the need to ensure that Ada compilers detect as many errors as possible. Since this need is not limited to developers of such systems, these requirements[2] were placed in this Chapter (dealing with general requirements), leaving the Chapter on safety-critical and trusted systems to deal with more specialized requirements. The first requirement deals with compile-time issues, and the second, with limiting the effect of programmer errors at run-time.

Revision requests associated with improving compile-time detection of errors fell into two categories — the reporting at compile-time of potential run-time errors such as exceptions that must be raised, and the rejection of compilation units containing unrecognized or defective pragmas. Both kinds of requests are met by this requirement.

RR-0165 Allow parameter constraint violations to be compile-time errors
RR-0209 Require the compiler to report certain-to-be-raised exceptions
RR-0242 Require compilation warnings for potential run-time errors
RR-0244B Flag run-time errors at compile-time when possible
RR-0261 Need compile-time warnings for access before elaboration errors
RR-0616 Require compilers to diagnose statically-detectable constraint errors

The following requests refer to the treatment of invalid pragmas. The requirement generally goes further than the RRs since it requires a compilation mode in which a compilation is rejected if a pragma is found to be invalid or unrecognizable.

RR-0211 Require compilers to report unrecognized or incorrect pragmas
RR-0692 Allow implementation-defined pragmas to cause unsuccessful compilation if restrictions implied by the pragmas are not obeyed
RR-0754 Require warnings for unrecognized pragmas
RR-0756 Require warnings when pragmas are ignored
AI-00850 Rejecting a unit when a pragma's assumptions are not met
This AI is being actively considered by the Ada Rapporteur Group.

[2]The lack of an initial capital ''R'' means we are neutrally referring to one of the topics to be addressed by the Mapping/Revision Team without distinguishing whether it is classified as a Requirement or as a Study Topic. This convention was also used in the Requirements Document itself.

Requirement R2.3-A(2) — Limit Consequences of Erroneous Executions

Unlike the previous requirement, this requirement is not presented as a study topic because it is phrased weakly (and hence, is easily satisfied) and because it is clear that some corrections can and should be made to the standard to restrict the allowed effects of erroneous executions under some circumstances.

Two of the revision requests associated with this requirement mention the difficulties caused when a task terminates silently because of an unhandled exception. Although such behavior is not, strictly speaking, erroneous, such run-time behavior leads to effects that can be hard to diagnose. These revision requests are categorized under this requirement because the requirement deals with run-time programmer errors, and the user need is to minimize the consequences of programmer errors. While it is not clear what can be done, solutions to other requirements might help mitigate this problem. For example, if finalization is provided in Ada 9X, the finalization code for a task might log the reason for its termination.

RR-0490 points out some of the ways that improperly written machine code insertions can lead to unpredictable program executions even though the standard does not mention such insertions as a source of erroneous program executions. The RR goes on to suggest that restricted forms of exception handlers be allowed in machine code insertions to facilitate recovery from exceptional situations detected in machine code insertions.

RR-0042 Clarify the meaning of incorrect-order dependence and its effects

RR-0066 Reduce risks associated with erroneous execution/incorrect order dependences

RR-0400 Do not allow a task to die silently on an unhandled exception
It is not clear what can be done, but the RR does point out a problem.

RR-0407B Do not allow a task to die silently on an unhandled exception
It is not clear what can be done, but the RR does point out a problem.

RR-0490 Need successful/convenient recovery from exceptions in machine code insertions
This simple request might improve safety of use of machine code insertions.

RR-0763 Allow nested scopes to turn off pragma SUPPRESS
Enforcing conventions for the correct use of pragma SUPPRESS can be important. The best solution, however, is not necessarily a pragma that turns off the effect of the pragma in a nested scope.

AI-00832 Effect of depending on parameter passing method when calling non-Ada programs

AI-00873 Type conversion/qualification of undefined scalar values
The AI suggests a simple upward-compatible rule change that is consistent with the intent of the Requirement.

2.4 Controlling Implementation-Dependent Choices

User Need U2.4-A: Uniformity of Compiler Behavior

Requirement R2.4-A(1) — Minimize Implementation Dependences

The following requests indicate various areas in which the language might be improved to remove possibly unnecessary implementation-dependent behavior.

RR-0007 Default representation for enumeration types should be specified
The representation for predefined type BOOLEAN should continue to be implementation-defined for efficiency reasons.

RR-0045 Allow/require extended precision for intermediate integer results
Extended precision is already allowed, but not required, by Ada; see 11.6(6) of the Standard.

RR-0061 Make Long_Float and Short_Float required types

RR-0068 The Standard should explicitly acknowledge that I/O support is optional for embedded systems
Implementation-dependent support for I/O functionality, particularly for implementations targeted to embedded systems, needs more attention.

RR-0187 Need to allow unsigned enumeration representation specifications
This RR suggests that the representation of enumeration values cannot be controlled adequately since the treatment of sign extension for negative literals is not adequately controlled by the standard. It is not clear that this complaint is justified, but it should be given consideration.

RR-0236 Reduce implementation-dependent behavior, or at least, ensure it is documented whenever possible

RR-0287 Make access types point directly to designated object
In some implementations, access values point to dope vectors rather than the designated object. This causes unnecessary implementation-dependence when interfacing with other languages.

RR-0302 The language should define literals for values of type ADDRESS

RR-0315 Allow integer type names that indicate representation size, e.g., INTEGER_32, to improve portability
This RR also recommends that the standard state explicitly that the length of LONG_INTEGER not be less than the length of INTEGER, with similar constraints imposed on SHORT_INTEGER.

RR-0353 Unchecked conversion should eliminate compiler-dependent fields
The RR points out an important problem in dealing with unchecked conversion, although the proposed solution is not necessarily the correct one.

RR-0355 Standardize means of getting the OS command line arguments
At the very least, compilers running under the same operating system should have the same way of interacting with command line arguments. This RR makes an interesting proposal on how to achieve this effect.

RR-0365 Reduce allowed variations in implementations to increase portability
RR-0432 is an expanded version of this RR.

RR-0432 Severely limit implementation options to improve portability
This RR gives a very extensive list of sections in the Standard that allow implementation-dependent choices to be made.

RR-0459 Improve support for interoperability; lessen implementation dependence
This RR lists several areas for consideration: representation clauses, the effect of pragma PACK, the effect of unchecked conversion, and permissible optimizations. An extensive and helpful discussion is provided.

RR-0709 Need more portability in getting command line inputs
See RR-0355 for specific suggestions.

RR-0732 Clarify semantics of instantiating ENUMERATION_IO with an integer type

There were two requests to specify the rounding behavior for half-integers and one request to at least document the behavior.

RR-0213 Need to be able to find out if an implementation rounds up or down

RR-0409 Define in the language how 3.5 rounds to integer

AI-00526 Rounding up or down

Two requests asked to make record representation clauses less implementation-dependent in their interpretation.

RR-0137 Standardize bit storage/order conventions

RR-0411 Express record representation clauses in a machine-independent way

3 Requirements for International Users

3.1 International Character Sets

The requirements specified in this section of the Requirements Document reflect extensive interactions with the international community via the Character Rapporteur Group (CRG), a subgroup of the ISO Working Group responsible for the Ada standard.

User Need U3.1-A: International Character Set

The following revision requests are reflected in each of the requirements stated in this Chapter.

RR-0330 Allow national characters in literals, comments, and identifiers
This request is addressed by Requirements R3.1-(A1-5).

AI-00510 Use ISO symbols and standards in the Ada ISO Standard
This commentary also requests that national alphabets be usable in identifiers, character literals, string literals, and comments. All of these requests are addressed by Requirements R3.1-A(1-5).

Requirement R3.1-A(1) — Base Character Set

The precision of this requirement reflects a decision by the CRG. If the CRG should subsequently revise this decision, it is intended that the Mapping/Revision Team take this into account. The CRG gave specific consideration to the idea of defining STANDARD.CHARACTER as a 128-character subtype of a 256-character type but rejected this approach because it was strongly desired that the predefined STRING type allow string literals drawn from a 256-character set.

The following revision requests are relevant to this requirement.

RR-0034 Ada should use ISO 8859/1-9 (8-bit) character set

RR-0148 Provide support for extended and graphic characters (256 ASCII set)

RR-0311 Generalize character set for 8-bit characters

RR-0367 Need support for national language character sets, including string comparison
The request for string comparison operations was not accepted, for reasons given in the Requirements document in the discussion following the requirement.

RR-0390 Need 8-bit unsigned CHARACTER for Greek and graphics symbols

RR-0736 Need 8-bit ASCII in Ada

AI-00420 Allow 256 values for type CHARACTER

Requirement R3.1-A(2) — Extended Graphic Literals

The following revision requests are reflected in the requirement for providing extended character set types.

RR-0050 Provide multi-national and multi-byte characters

RR-0331 Need predefined LONG_CHARACTER (16 bits) and LONG_LONG_CHARACTER (32)

RR-0438 Allow use of multi-octet character set

Requirement R3.1-A(3) — Extended Character Set Support

Although no revision requests were submitted calling just for support of input/output operations using international character sets with more that 256 graphic symbols, this requirement reflects other communications from the international community and RR-0330.

3: Requirements for International Users

Requirement R3.1-A(4) — Extended Comment Syntax

As part of a complete solution to the use of Ada by non-English-speaking programmers, the ability to write comments in the programmer's native language is important. Although this requirement is already, in principle, satisfied by the current standard (see AI-00339), it was stated explicitly in the Requirements Document lest it be thought to have been excluded intentionally.

Study Topic S3.1-A(5) — Extended Identifier Syntax

The CRG has explicitly asserted that it is not desired to translate reserved words. In particular, although it may be helpful to have unreserved identifiers rendered in an extended character set, representatives of the international community have indicated that translation of reserved words is not desired because it would impair the interchangeability of programs. The requirement implies that reserved words should not be translated because it states that Ada's syntax and lexical rules must be kept independent of the source code representation.

4 Support for Programming Paradigms

4.1 Subprograms

User Need U4.1-A: Dynamic Subprogram Selection

Study Topic S4.1-A(1) — Subprograms as Objects

The programming paradigm known as object-oriented programming has become much more widely known since Ada was designed. One of the key issues in formulating the Ada 9X requirements was the extent to which Ada 9X should support the object-oriented paradigm. In discussions during the requirements phase, it became clear that it would be a mistake to use the term ''object-oriented'' directly in any requirement because there was insufficient consensus on what language capabilities were implied by the term. Hence we developed requirements that specified capabilities consistent with some views of object-oriented programming without directly using the term. The ability to treat subprograms as values of variables is one such capability that can be used to provide functional capability equivalent to the kind of polymorphism provided in some object-oriented programming languages (e.g., by writing Obj.Func(Obj, …)). For full polymorphism, Ada's strong typing rules must be relaxed in some way (see Study Topic S4.3-B(1)). Several revision requests directly called for subprogram variables.

RR-0081 Provide subprogram and package types
Subprogram types is an obvious solution to this requirement. The RR is very short and provides no arguments explaining the need for package types.

RR-0430A Need objects of a subprogram ''type''

RR-0441 Extend Ada to allow for polymorphism

RR-0503 Provide subprogram types for dispatcher-style programming
This RR was particularly useful in formulating the associated User Need and requirement.

RR-0563 Need to allow subprogram types and variables

RR-0611 Allow subprogram types, variables, constants, parameters, etc
This RR was helpful in formulating the associated User Need and the requirement.

RR-0647 Need ability to select actions depending on state without using case statements

User Need U4.1-B: Interfacing with Non-Ada Subprograms

Requirement R4.1-B(1) — Passing Subprograms as Parameters

Since the requirement for subprogram variables was classified as a Study Topic, and since the ability to pass subprograms as parameters to non-Ada procedures and functions was clearly needed and could be supported with a simpler mechanism than subprogram variables, a separate Requirement was written to cover this need. If it proved too difficult to support subprogram variables directly in Ada, at least Ada 9X would provide a standard way of passing subprograms to non-Ada procedures and functions.

RR-0064 Allow some form of subprogram callback

RR-0128 Provide subprograms as parameters to subprograms and entries

RR-0180 There is a need for procedures as parameters for X-Windows, etc

RR-0388 Proposal for clean way of executing a subprogram by its address
A straightforward subprogram type provides a simpler solution than the approach proposed in this RR.

RR-0414 Ada needs subprogram types and subprogram objects

RR-0422 Allow subprograms as parameters and maybe also as values

RR-0430B Need to pass subprograms as parameters

RR-0512 Provide subprograms as parameters to subprograms
This RR gives some examples of the limitations of using generic parameters as a means of getting the effect of passing subprograms as parameters.

RR-0629 Need procedure and function types for use in subprogram calls

RR-0641 Add subprograms as parameters to the language

RR-0774K Allow subprograms as parameters

Requirement R4.1-B(2) — Pragma INTERFACE

The Ada 9X study report on implementation-dependent pragmas and attributes [7] showed that it would be helpful to impose a greater degree of compiler uniformity on the functionality associated with pragma INTERFACE. Although this requirement could have been subsumed under Requirement R2.4-A(1) (Minimize Implementation Dependences), it seemed appropriate to mention it in a section dealing with subprogram requirements.

RR-0014 Need to call subprograms loaded in ROM
Ada 9X could ensure that pragma INTERFACE is usable to identify subprograms located in ROM.

RR-0527 Standardize information/conventions used for pragma INTERFACE

4.2 Storage Management

User Need U4.2-A: Control of Storage Use

Requirement R4.2-A(1) — Allocation and Reclamation of Storage

Various revision requests point out how Ada programmers are not necessarily given sufficient control over storage usage, and in particular, how some implementations do not take sufficient care to recover unused storage.

RR-0112 Provide user support for controlled space reclamation
This RR provides an example user interface for controlling storage allocation and reclamation.

RR-0113 Ensure that there are no storage ‘‘leaks’’
This RR gives some examples of how storage leaks can occur.

RR-0118 Provide a user-specified storage reserve for STORAGE_ERROR recovery
This capability could be automatically made available if Ada 9X allowed user-defined storage management operations to be written.

RR-0120 Allow users to defer the signalling of STORAGE_ERROR when space is exhausted
The problem can be solved in its full generality only by customizing a storage allocator.

RR-0370E Need to recover space for task control blocks when tasks are created by an allocator

RR-0374 Ada should address memory management requirements in distributed systems

RR-0439 Require automatic garbage collection

RR-0493 A programmer should be able to ensure that storage will be reclaimed

RR-0643 Garbage collection can now be done well; encourage its use
Section 4.2 of the Requirements Document addresses this topic further.

RR-0702 There is a need for improvements in heap storage management

RR-0774A Make it possible to write NEW in Ada

Study Topic S4.2-A(2) — Preservation of Abstraction

User-defined assignment was almost provided in the original design of Ada. The design team, however, did not take this step because there was insufficient time to be sure that its inclusion would not lead to implementation inefficiencies or other anomalies. An effort was made, however, to make it as easy as possible to allow user-defined assignment operations when the language was revised.

The Requirements Team could have articulated a general requirement for capabilities allowing user-defined types to be indistinguishable from pre-defined types. Such a requirement would have entailed the ability to specify user-defined basic operations, including assignment, conversion, and various attributes. The Team did not develop such a requirement because it might have required a more complicated language change than would have been justified by Guideline **G-1**. However, it was clear that the ability to control the use of storage was very important in many applications, so the Team instead developed a requirement that addressed primarily this need. It was stated as a Study Topic because of misgivings over the possible scope of changes that might be needed; we made a conscious decision that if the need for user-defined storage management could not be supported with minimum disruption to the language and implementations, we would be willing to live without the capability. Since the ability to recover unused storage requires the ability to get control when frames are being exited, this requirement would also allow the concept of finalization to be supported by the revised language. This concept, of course, has greater utility than merely supporting user-defined storage management, but given our concerns over the ability to support finalization with no costs if it was not used, we were unwilling to mention it directly in the requirement and were, in any event, unwilling to give it great urgency.

Several revision requests asked for user-defined assignment and finalization in conjunction with storage management issues, while others just asked for the capabilities on general grounds of expressive power and support for programming abstraction.

RR-0001 Limited types need assignment, constants

RR-0070 Allow user-defined assignment for limited types

RR-0088 Problems associated with user-defined assignment
This RR points out some problems to be addressed if user-defined assignment is added to the language.

RR-0160 Allow user-defined assignment for limited types

RR-0184 Need user-defined assignment operator for limited private type

RR-0201B Overload the assignment operation

RR-0202 Relax parameter mode rules for limited types that have an assignment operation
These problems will be addressed by allowing user-defined assignment for limited types.

RR-0413 Allow user-written := for all types

RR-0515 Need ability to request indivisible update for specific objects, especially in distributed systems
The submitter objects to the need to explicitly program mutual exclusion when making assignments to specific objects, and would like to have the assignment operation imply indivisible update. This capability could be provided by user-defined assignment.

RR-0541 Allow user-defined :=, =, DESTROY operations to support memory management
This RR gives a very lengthy discussion and examples showing why user-defined assignment and finalization are needed to provide appropriate memory management functionality under user control.

RR-0544 Need indivisible update on reference counts
This RR briefly discusses the difficulties of maintaining reference counts for data shared among tasks. It may provide an interesting example to use when evaluating Ada 9X solutions.

RR-0609 Allow user-defined override of =, /=, := on all types

RR-0663 Allow certain overloading of := and subscripting
User-defined subscripting is not required.

RR-0669 Allow user-written := routines

Other revision requests asked for a ‘‘finalization’’ capability, which is also needed to control the use of storage safely.

RR-0003 Provide a compiler-independent finalization mechanism
See the discussion following the Study Topic.

RR-0019 Allow types to specify finalization procedures for safely controlling use of collections

RR-0092 Allow user-specified finalization

RR-0168 Allow implicitly-invoked finalization code for storage management

RR-0203 Allow finalization code for packages and tasks

RR-0385 Need finalization code for packages

RR-0466 Allow user-defined finalization for objects of a type to ensure release of resources

RR-0475 Need automatically-invoked user-defined routines to reclaim storage

RR-0523 Allow user-defined finalization for objects of a type to ensure release of resources

RR-0660 Need constructors and destructors for package types

RR-0676 Add finalization to ensure release of resources

RR-0774N Allow task cleanup on termination of parent
Finalization is one of the matters to be studied.

4.3 Composition of Program Units

User Need U4.3-A: Reduced Recompilation

Study Topic S4.3-A(1) — Reducing the Need for Recompilation

Various RRs give examples of language rules that lead to an increased need for seemingly unnecessary recompilation.

RR-0142 Reduce cases where recompilation of subunits is needed
This RR gives some examples of the kinds of program changes that should not force recompilation.

RR-0307 Allow completion of private declarations to be in the package body
The RR gives a reference to a paper justifying a conclusion that efficient code can be generated even if a private type’s full declaration is given in a package body.

RR-0368A Ensure unnecessary recompilation is avoided

RR-0451 Changes to package constants should not cause recompilation

RR-0688 Unnecessary recompilation required when redeclaring a subprogram body
This RR gives an example where recompilation should not be required.

User Need U4.3-B: Programming by Adaptation of Existing Units

Study Topic S4.3-B(1) — Programming by Specialization/Extension

The ability to extend packages without modifying the original package (this is an example of what is called ‘‘inheritance’’ in object-oriented terminology) can reduce the need for recom-

pilation (see Study Topic S4.3-B(1)). To avoid preconceptions about object-oriented programming mechanisms, this study topic simply states one of the significant capabilities provided by inheritance mechanisms. In this way, the requirement focuses on the functional capability that is needed.

Some revision requests asked directly for inheritance mechanisms:

RR-0125 Introduce object-oriented inheritance into the language

RR-0140 Provide support for object-oriented programming

RR-0167 Allow classes of abstract data types

RR-0223 Need to add inheritance to support object-oriented programming

RR-0440 Extend Ada to be truly object-oriented

RR-0442 Extend Ada to allow a package type hierarchy

RR-0516 Provide more support for object-oriented programming

RR-0525 Extend Ada to allow for polymorphism and inheritance

RR-0662 Need package classes and inheritance for object-oriented programming

RR-0668 Need package types to get, for example, an array of packages
The requirement provides much of the requested functionality.

RR-0750 Add support for inheritance and polymorphism to the language

Other requests noted special problems that should be considered in developing a response to the Study Topic. In particular, several requests were concerned about the ability to develop operations for types declared in one or more packages.

RR-0052 Multiple derived types from same package do not give desired operations
RR-0482 contains a better example and motivation.

RR-0482 Multiple derived types from same package do not generate needed operations
Specialization/extension facilities should help solve this problem.

RR-0533 Mutually recursive types from different packages cannot be done
This is an example of a problem that might be solvable with suitable facilities for specializing/extending packages and types.

Some requests wanted the ability to separate implementation-dependent aspects of a package from implementation-independent aspects. This requirement might well be met with an appropriate specialization/extension capability.

RR-0065 To improve reuse possibilities, allow rep clauses and various pragmas to be separated from the compilation unit to which they apply

RR-0171 Allow target-dependent code (including rep clauses) to be separate from other code

RR-0698 Need ability to separate portable and non-portable code into separate units

Some requests noted the need to extend the representation of a private type. This is a special need that falls under the stated requirement.

RR-0560 Need to access a private type's representation in related packages

RR-0684 Related packages need access to a private type's representation

Finally, there were a few requests that reflected the Study Topic fairly directly.

RR-0069 Allow subprograms and types to be added to a package without modifying the original package

RR-0172 Make import and export of types easier

RR-0174 Allow packages to be generic with respect to concurrency protection
This is a good example of a form of specialization that is often needed.

RR-0448 Allow different sets of subprograms to depend on common declarations

RR-0455 The import and export mechanisms of Ada are too limited
This RR contains a fairly extensive discussion of problems that are relevant to the Study Topic.

RR-0505A Provide extendable record types

RR-0531 Variants of a type can't be usefully supported with current variant record approach
Although this RR discusses variant record limitations, the real problem being addressed is the ability to construct efficiently representable alternative representations for a conceptual type. The RR mentions explicitly that object-oriented languages allow this kind of problem to be solved more straightforwardly.

RR-0540 Allow a new package to build on an existing package

RR-0599 Certain changes to derived/private types will help inheritance
This RR gives some generic examples of difficulties of extending existing units to meet new needs.

User Need U4.3-C: Library Support

Study Topic S4.3-C(1) — Enhanced Library Support

These revision requests discuss various ways in which Ada's support for libraries might be improved.

RR-0091 Don't specify the compilation process in the Standard

RR-0177 Standardize interface between compiler and library for configuration management

RR-0226 Need standardized support for improved library management capabilities
Presents a reasonable discussion of library management needs.

RR-0237 Make separate compilation independent of a particular library model

RR-0283 Need convenient way to set global compilation parameters

RR-0368B Ensure the library can be manipulated by tools other than those provided by the compiler vendor

Quite a few RRs asked for more flexibility in restricting the visibility of library unit names.

RR-0073 Allow visibility of names to be restricted within a program library

RR-0178 Problems with name clashes with big program libraries

RR-0457 Structure library units as groups, control visibility of library units

RR-0774C Extend control of library unit visibility

Finally, there were many RRs that asked to relax the rule requiring that subunit names be distinct within a given subunit hierarchy.

RR-0038 Allow expanded instead of simple names of subunits to be distinct

RR-0041 Allow overloaded subunits with respect to a common ancestor library unit

RR-0402 Need unique hierarchical pathnames for subunit

RR-0557 The use of renaming declarations to provide subprogram bodies helps get around the inability to overload subunit names

AI-00458 Problem with naming of subunits

4.4 Generics

User Need U4.4-A: Support for Reusability

This User Need deals primarily with improving the use of generic units.

Study Topic S4.4-A(1) — Generic Formal Parameters

This requirement was classified as a Study Topic because it is unclear to what extent changes to generic formal parameters are needed given Study Topic S4.3-B(1). Moreover, it is unclear whether the proper solution to the needs reflected in the revision requests is to allow additional forms of generic parameters or simply to relax the matching rules, thereby allowing a wider use of formal parameters within a generic unit even though some instantiations may then become illegal.

The following revision requests essentially asked for a weakening of the generic contract model.

RR-0458 Need convenient way to escape into weakly typed subprogram call

RR-0713 Relax array matching rules for generics

Several requests asked for the ability to pass an exception to a generic unit so the unit could raise the exception or so an exception raised by a formal subprogram could be handled within the unit.

RR-0033B Need to pass exceptions to subprograms and generic units

RR-0101B Need to pass exceptions as parameters to generic units and subprograms
Most of this RR deals with the ability to group exception names.

RR-0228 Allow generic parameterization with exceptions

RR-0383 Need generic exceptions for truly reusable generic units

RR-0468 No generic way to handle exceptions raised by generic formal subprograms

RR-0526B Need to pass exceptions as parameters to generic units and subprograms

RR-0621B Permit exceptions as generic formals

RR-0671 Allow exceptions as generic parameters

RR-0706 Allow exceptions and packages as generic parameters

RR-0774J Allow generic parameters for any Ada entity, e.g., exceptions

Three requests dealt with the need to improve the treatment of staticness within generic units. See RR-0342 for concerns about how allowing a greater use of static expressions in generic units might impair the ability to share code for generic instances.

RR-0227 Allow generic parameterization with static numeric quantities

RR-0445 Non-staticness of generic formals poses problems
This RR argues that it should generally be possible to turn a non-generic unit into a generic unit, but this is not easily possible when the non-generic unit uses static expressions (in case statements, aggregates, and type declarations) that depend on formal parameters in the generic version.

RR-0712 Need ability to declare double precision numeric types within a generic unit
The problem here is mainly the inability to create new numeric types whose precision is a function of a generic formal type. Relaxing the rules concerning static expressions would help in the creation of numerical library packages.

Several requests noted that actual entry parameters could not be used in timed or conditional entry calls within a generic unit. AI-00451 points out that such calls are sometimes needed to avoid deadlocks.

RR-0408 There is a need for generic formal entries

RR-0486 Allow generic formal task types as well as generic formal limited types
This RR points out that if a generic unit expects a task as an actual parameter, the programmer is unable to express this requirement.

RR-0488 Allow generic formal entries as well as generic formal subprograms
This RR gives an example of how a generic formal entry would be used to achieve the effect of an asynchronous call.

RR-0659 Need to make entry call on a generic formal parameter

AI-00451 Task entries as formal parameters to generics
This AI discusses some workarounds that are needed if entries are not distinguished as generic formal parameters.

There were a number of requests to generalize generic units so record types could be handled. It is not clear that this is advisable, but this is for the Mapping/Revision Team to determine.

RR-0027 Improve generics so a generic report generator could be written

RR-0505B Allow partial match for records as generic parameters

RR-0627 Allow partial match to formal type for records

RR-0722 Need generic formal record types

AI-00452 Allow record types as generic formal parameters

These requests give useful examples showing why it is necessary and useful to be able to access numeric base types within a generic unit.

RR-0190 Allow use of a base type within a generic unit

RR-0511 Allow use of a base type within a generic unit

AI-00285 Need to be able to access a base numeric type in some algorithms
This AI gives an example of the difficulties of getting access to a numeric base type, which was needed when trying to write TEXT_IO.FIXED_IO in Ada.

AI-00291 Can't define a generic package that works for all floating point types

User Need U4.4-B: Independent Use and Implementation of Generics

Requirement R4.4-B(1) — Dependence of Instantiations on Bodies

There was only one revision request directly related to this requirement.

RR-0562 Require separate compilation of generic specifications and bodies

Study Topic S4.4-B(2) — Tighten the "Contract Model"

All but one of the revision requests relevant to this requirement are concerned about the fact that the legality of a generic instantiation can depend on the use of a generic formal parameter in the generic body, namely, an object of a generic formal type cannot be declared legally if the corresponding actual type is an unconstrained array type (e.g., STRING) or an unconstrained record type without default discriminants. RR-0584 notes additional possibility for error due to allowed mismatches between formal and actual parameter subtypes.

RR-0006 Distinguish unconstrained/constrained generic formal types

RR-0446 Tighten the contract model by distinguishing constrained/unconstrained generic types

RR-0472 Distinguish unconstrained/constrained generic formal types

RR-0549 Ensure the use of unconstrained actual types is always legal

RR-0584 Need stricter checking of formal generic subtypes when an instantiation is given
This RR provides a careful discussion of an error-prone aspect of generics, namely, the fact that formal subtypes are sometimes ignored in matching actual parameters.

User Need U4.4-C: Code Space Compactness

Requirement R4.4-C(1) — Generic Code Sharing

The relevant revision requests point out some difficulties in supporting generic code sharing.

RR-0005 Exception declarations in generic packages make code sharing unnecessarily difficult

RR-0342 Do not implement requests that will break generic code sharing
This RR discusses how changes in the rules for treatment of static types and expressions could impair generic code sharing possibilities. The RR discusses the potential effect of RR-0027 and RR-0048.

RR-0585 Need pragma to specify code-generation strategy for generic instantiation

RR-0586 Different instantiations of the same generic unit may have to evaluate their actual parameters in different orders
The RR asserts that for a stack machine, this causes inefficiencies for shared-code generics.

4.5 Exceptions

User Need U4.5-A: Exception Name

Requirement R4.5-A(1) — Accessing an Exception Name

There were many revision requests that asked for the ability to get the name of a raised exception while executing a handler for that exception. Some of the requests asked for additional contextual information as well. The Requirements Team decided that the name of a raised exception could be provided easily by implementations, but that to go any further would probably incur more costs in language and implementation complexity than would be worthwhile.

RR-0033A Need to find the name of a raised exception

RR-0085 Need to get the name of the current exception
This RR gives some valid reasons for accessing the name of an exception.

RR-0145 Provide a way to get exception name from WHEN OTHERS handlers
This RR references a POSIX requirement for the ability to print the name of an exception from within an OTHERS handler.

RR-0219 Provide a way to get the name of the last raised exception, including an out-of-scope exception

RR-0403 Need to be able to get the name of the current exception

RR-0407A Need exception name, line number, and unit name where raised
This RR cites a requirement for logging exception information in information systems. The 9X requirement, however, does not go so far as to request contextual information such as the name of the compilation unit, source code line, etc. The requirement allows additional information to be made available if this can be done with little implementation cost.

RR-0477 Provide a way to get the name and location of a raised exception
Obtaining traceback information is not part of the 9X requirement.

RR-0526C Need to determine the name of a raised exception

RR-0582 Provide standard interface for getting additional implementation-dependent info about state when an exception is raised
The requirement allows additional information to be made available if this can be done with little implementation cost.

RR-0621A Need to find out which exception has been raised

RR-0772 Need to be able to get exception name in a handler

RR-0774E Provide access to context of an exception situation
The requirement allows additional information to be made available if this can be done with little implementation cost.

RR-0774G Provide exception name in OTHERS handler

In addition to requests for the name of a raised exception, there were several requests asking for a way to group exceptions; the requests were not included in a 9X requirement because such a change was not considered to provide enough user benefit to justify disturbing the language, given other changes already reflected in the requirements. Two requests asked for finer granularity in the predefined exceptions, but this issue was given sufficient consideration in the original design and the revision requests did not provide sufficient rationale for re-examining this decision.

RR-0036 Allow exceptions to be grouped under a single name by allowing exception subtypes

RR-0101A Allow exceptions to be grouped under a single name

RR-0416 Granularity of predefined exceptions is too coarse
This issue was given thorough consideration in the original design, and insufficient evidence is given in this RR to justify reconsidering the decision.

RR-0526A Allow exceptions to be grouped under a single name

RR-0774H Provide more predefined exception names with finer granularity
This issue was given thorough consideration in the original design, and insufficient evidence is given in this RR to justify reconsidering the decision.

4.6 Input/Output

User Need U4.6-A: Interactive Text Input/Output

Requirement R4.6-A(1) — Interactive TEXT_IO

The requirement and the associated revision requests point out potential areas for improvement in the TEXT_IO package to support interactive I/O more conveniently. The following RRs should be considered in responding to this requirement.

RR-0047 Add TEXT_IO.GET functions
There might be some benefit in this suggestion to add value-returning subprograms to TEXT_IO given that a variable length string package is uniformly available.

RR-0149 Provide a keyboard input/output package

RR-0164 Provide multitasking terminal I/O in TEXT_IO

RR-0235 Need support for interactive terminal input/output

RR-0597 Need functional version of GET_LINE instead of procedural
RR-0047 gives a stronger justification for this change.

AI-00485 Having independent standard input and output files is not useful for interactive I/O

AI-00488 Skipping of leading line terminators in GET routines causes problems in interactive I/O

One RR related to interactive input-output goes beyond the intent of the requirement by requesting a special package of screen management functions. This request was not accepted.

RR-0089 Provide facilities for I/O screen operations
Although there is a requirement for improved interactive text I/O functions, this request goes too far by asking for a screen management package. Text I/O is only intended to provide common, basic functionality.

User Need U4.6-B: Various Input/Output Functions

Requirement R4.6-B(1) — Additional Input/Output Functions

Several requests asked for simple functional extensions:

RR-0207 Add TEXT_IO support with Exists function and Append procedure

RR-0382 Need to be able to rename and append to a file in standard Ada

RR-0404 Need convenient way to find out if a particular file exists

RR-0405 Need convenient way to append to a file

RR-0420 Need file ''extend'' or ''append'' capability

Two requests note that the definition of GET_LINE makes it inconvenient to use when the buffer length is equal to the line length, since the final page and line terminator is skipped automatically in this case. It would be more consistent to most users if automatic skipping only occurred for initial terminators, as for other TEXT_IO operations, although one comment notes that skipping leading terminators can also be inconvenient for interactive I/O (see AI-00488).

RR-0553 GET_LINE should not automatically call SKIP_LINE

AI-00605 GET_LINE skips terminators at the end of the line, which is inconsistent with other GET procedures

Two requests noted an inconvenience in the way the default formats for INTEGER_IO, FLOAT_IO, FIXED_IO, and ENUMERATION_IO are controlled by a programmer. RR-0484 proposes a simple upward compatible improvement.

RR-0130 Replace DEFAULT_xy variables in Chapter 14 by functions
RR-0484 proposes a better change.

RR-0484 Add DEFAULT_xy functionality as parameters to generic TEXT_IO packages
This RR proposes a simple, upward-compatible solution that improves the usability of the numeric IO packages.

The remaining requests cover a range of possible functional improvements.

RR-0159 Add standard package of general file system functions

RR-0295 Create TEXT_IO.PUT_LINE for types other than string (make like PUT)
The operations called for here arguably improve the uniformity and teachability of TEXT_IO, but might also be considered to clutter the definition.

RR-0359 Allow mixed case output for enumeration literals

RR-0361 Increase the number of options for controlling the output format of numbers

RR-0447 Need to be able to preserve/restore the default file at any point
This RR provides a useful argument for the requested capability.

RR-0485 Provide means to get the line length of an input or output device

RR-0551 Need assignment capability for TEXT_IO.FILE_TYPE
See RR-0447 for a workaround that can be used today.

RR-0593 Mandate implementation of variant record I/O in DIRECT_IO/SEQUENTIAL_IO
The required functionality is available in principle by using ''shared files'', as suggested in the discussion following Requirement R4.6-B(1).

RR-0711 I/O by a task in multi-task application should not block whole program

AI-00329 Look-ahead operation for TEXT_IO

AI-00345 Record type with variant having no discriminants
The principle requested use for untagged variants is for I/O.

AI-00487 END_OF_PAGE and END_OF_FILE should not return TRUE when there is still an empty line to be read

5 Real-Time Requirements

5.1 Time

User Need U5.1-A: Time Measurement

Requirement R5.1-A(1) — Elapsed Time Measurement

The requirement for measuring elapsed time stems mainly from interactions with the real-time community, e.g., as documented in [5]. The revision requests listed here refer mostly to the ability to ensure that applications have sufficient control over clock precision. These RRs should be considered in evaluating proposed solutions to the timing requirements.

RR-0105 Allow application to set/adjust clocks
The ability to adjust the elapsed time or time-of-day clocks is implicit in the stated requirement.

RR-0107 Allow application to specify clock timing interval if hardware allows this flexibility

RR-0276 Need user specified accuracy and precision control over timing

RR-0280 Short delays are too inefficient; Calendar time unnecessary; timing performance must be documented
The general tenor of this request is that Ada's timing model is not appropriate for embedded real-time systems, but the purpose of the requirements is to ensure that the Ada 9X model indeed is appropriate.

AI-00223 Require adequate resolution for the function CLOCK

User Need U5.1-B: Periodic Computation

Requirement R5.1-B(1) — Precise Periodic Execution

Several revision requests support the stated requirement.

RR-0108 Need to be able to wake up a task at a particular local time
Provides a good discussion of some of the issues.

RR-0306 Need to be able to start processing at a particular time of day

RR-0352 Require Calendar.Clock to return consistently accurate local system time
A real-time annex should specify constraints on timing accuracy.

RR-0410 Provide explicit language support for periodic tasks
This RR goes beyond the requirement since it requests direct language support for specifying task periodicity. The arguments should be considered, however, in evaluating Ada 9X proposals.

User Need U5.1-C: Overrun Detection and Response

Requirement R5.1-C(1) — Detection of Missed Deadlines

The need to deal with missed deadlines arose mainly from interactions with real-time users, but there were a few revision requests that stated this need.

RR-0384 Cannot write subprogram which causes an exception after specified delay

RR-0656 Need timed exceptions for deadline scheduling
The requirement should provide the requested functionality.

5.2 Task Scheduling

User Need U5.2-A: User-Controlled Scheduling

Requirement R5.2-A(1) — Alternative Scheduling Algorithms

There were many revision requests that, in essence, asked for greater control over run-time system decisions. Two requests explicitly asked for the definition of a standard run-time system interface as the solution to this need.

RR-0074 Define a standard run-time support environment interface

RR-0175 Define interface between compiler- and target-specific run-time system aspects
The possibility of a standardized RTS interface is discussed under this requirement.

Several requests wanted to ensure that task scheduling and the use of hardware resources by the run-time system could be completely controlled by an application.

RR-0016 Allow user-selectable task scheduling algorithms

RR-0037 Allow tasks (i.e., delays) to execute using simulated time rather than a real-time clock
An application-controlled scheduling interface might allow tasks to execute in simulated time, although such a capability goes beyond the intent of the stated requirement.

RR-0121 Provide more user control over scheduling decisions

RR-0124 Ensure that code dependent on task scheduling algorithms is portable
Although this RR discusses AI-00594, which is not an approved AI, the concern of the RR is reflected in its title. This concern has been addressed by the notion of a real-time Annex for Ada 9X, since one purpose of the annex is to improve the performance portability of code. In addition, the requirement for user-controlled scheduling algorithms makes portability more possible.

RR-0170 Permit or provide alternate scheduling algorithms

RR-0286A Embedded system users need the ability to control timer utilities

RR-0286B Embedded system user may need access to interrupts that are also used by the run-time system

RR-0286C Run-time system should avoid entering privileged mode

RR-0379 Application should select the specific scheduling algorithm

Other requests dealt with specific schedulability issues. Some requests were concerned with the ability to specify task priorities in a more flexible fashion. Although most of the requests were concerned with changing priorities at run time due to mode changes, one RR also noted that requiring the static specification of a task's priority was unhelpful during system integration or reconfiguration (see RR-00654) since a task's priority might be determined at load time.

RR-0020 Relative importance of functions may change during program execution, so priorities should be changeable

RR-0116 User-modifiable priorities needed for mode change and graceful degradation
This RR gives brief examples supporting the stated need.

RR-0192 Need ability to change priorities during mode change and for graceful degradation

RR-0337 Provide some form of user-modifiable priorities
Both mode changes and graceful degradation are mentioned in examples.

RR-0347 Allow applications to change priorities under program control; allow task priority to increase as a function of lack of service

RR-0370D Need to set priorities of tasks during mode shifts

RR-0654 Need non-static priorities
This RR notes, in effect, that the appropriate priority of a task depends on what other tasks are executing on the same processor, and this might change during system development, maintenance, or run-time configuration, in which case, the language is too restrictive in requiring that priorities be specified with static expressions.

Several requests were concerned with the use of priorities in determining which task to select from an entry queue or from open alternatives of a selective wait. Some requests asked that Ada 9X support the special scheduling algorithm known as priority inheritance.

RR-0015 Allow task priorities to control all queuing/select decisions

RR-0021 Need priority inheritance for server tasks

RR-0072 Prioritized queues and priority inheritance are needed for real-time applications

RR-0075 Queue entries by task priority or FIFO based on application

RR-0076 Allow selection of entry calls from entry queues and open alternatives based on priorities

RR-0193 Allow priority queues, priority inheritance, and prioritized treatment of open select alternatives

RR-0415 Allow priority inheritance, prioritized entry-queues, and prioritized selective wait

RR-0657 Order entry queues based on priority

RR-0737 Allow reliable user control over selection of alternatives in a select statement
The RR explicitly notes that the 'COUNT attribute is not sufficient to ensure the requested degree of control.

Requirement R5.2-A(2) — Common Real-Time Paradigms

Most revision requests asked for support for mutual exclusion that would be more efficient and more natural than the use of rendezvous:

RR-0084 Specify standard conventions for using tasks that permit high-performance implementations
The intention here is to specify, in a real-time annex, the restrictions on task usage that allow tasks used for mutual exclusion to be implemented with special efficiency.

RR-0185 General Ada rendezvous is slow; semaphores would be better

RR-0241 Need easier and more efficient support for mutual exclusion

RR-0278 Tasking model should support common scheduling disciplines more easily

RR-0461 Provide standard package of semaphore operations

RR-0521 Need more convenient support for use of shared memory among tasks

RR-0590 Need clear, efficient, standard support for mutual exclusion
This RR gives a detailed example of a problem that is to be solved by improved mechanisms in Ada 9X.

Three requests asked for the ability to combine entry calls and accept statements in a single selective wait.

RR-0498 Make selective wait symmetrical with respect to accept statements and entry calls

RR-0658 Allow accept statement possibility in a conditional entry call

RR-0697 Allow entry call alternative in selective wait
RR-0498 provides a rationale for this capability.

5.3 Asynchronous Control of Execution

User Need U5.3-A: Asynchronous Control of Execution

Requirement R5.3-A(1) — Asynchronous Transfer of Control

Several revision requests addressed the need for an efficient, user-controlled mechanism for terminating and restarting task executions.

RR-0063 Protect tasks from being aborted while performing critical functions
Although this request is phrased in terms of protecting a task from being aborted, the underlying need is met by the Ada 9X requirement.

RR-0083 Provide asynchronous transfer of control via entry call/selective wait construct
The proposed solution is attractive.

RR-0106 Provide asynchronous transfer of control
This RR contains a good rationale and good examples dealing with the need for asynchronous transfer of control.

RR-0196 Endorsement of RR-0083
This RR endorses the solution suggested in RR-0083 and repeats some material found in [8].

RR-0335 Effect of abort statement is too implementation-dependent
The requirement addresses the problem raised in the RR.

RR-0431 A terminate alternative cannot be used to stop cyclic tasks
An asynchronous transfer of control construct might serve to meet the need described here.

RR-0651 Allow one task to raise an exception in another task
A different solution to the problem is called for.

RR-0742 Need ability to asynchronously stop another task

RR-0768 Need to asynchronously interrupt another task to stop it

AI-00450 Should allow raising of an exception in another task
A different solution is called for.

5.4 Asynchronous Communication

User Need U5.4-A: Asynchronous Message Passing

Requirement R5.4-A(1) — Non-Blocking Communication

This requirement deals with intertask communication within a single program. There is also a need for interprogram communication. That need is addressed under Requirement R8.1-A(1).

RR-0183 Asynchronous inter-task communication is not available

RR-0587 Provide for communication between loosely coupled tasks

RR-0655 Add asynchronous message queues

RR-0665C Support message-driven intertask communication

RR-0748 Provide standard package of asynchronous primitives

User Need U5.4-B: Asynchronous Multicast

Study Topic S5.4-B(1) — Asynchronous Multicast

The need for this capability was established during reviews with real-time system developers. There was only one revision request that explicitly mentioned such a need.

RR-0665A Support multicast message transfer

6 Requirements for System Programming

6.1 Unsigned Integer Operations

User Need U6.1-A: Unsigned Integers

Requirement R6.1-A(1) — Unsigned Integer Operations

The need to deal with unsigned integer representations and typical operations on such representations was mentioned in several revision requests.

RR-0136 Provide support for bit-field operations such as shift, rotate

RR-0138 Need full-sized unsigned integers

RR-0139 Provide shift and rotate operations for boolean arrays
The requirement supplies the requested functionality.

RR-0188 Embedded applications need unsigned integers and bit-wise logical operations on integer types

RR-0332 Provide unsigned integer capability
This RR provides a fairly extensive discussion of the need and the language design difficulties.

RR-0433 There is a need for predefined unsigned integer types

RR-0460 Ada needs to provide support for unsigned integer types
This RR provides an extensive discussion of the issues and a detailed solution that helps to indicate the full range of the requirement.

RR-0633 Provide logical operations (e.g., XOR) for integers

RR-0634 Provide arithmetic shift operations for integers

RR-0640 Need to access chunk of a bit vector as a whole
The requirement provides much of the requested functionality.

RR-0721 Try to add unsigned integers to the language

RR-0766 Allow bit-wise operations (AND, SHIFT) on integers, bytes, etc

AI-00600 Why we need unsigned integers in Ada

6.2 Data Interoperability

User Need U6.2-A: Data Interoperability

Requirement R6.2-A(1) — Data Interoperability

Two requests are concerned with the ability to control how data is blocked for efficient transmission. The first (RR-0103B) is concerned with breaking up a large data structure into smaller blocks to reduce I/O buffer size, and the second (RR-0773) is concerned with grouping several variable length records into a single block for efficient data transmission.

RR-0017 Be able to treat an Ada object as an array of storage units
This RR gives an extensive example and discussion of difficulties in writing I/O packages for arbitrary data types.

RR-0103B Provide efficient means of reading large data structures in chunks
This problem could also be solved by providing an appropriate FORM parameter when opening a file, so a large data structure would be read or written in several blocks, thereby using smaller internal buffers.

RR-0289 Need multiple views of a record structure even when no discriminant is present
Unchecked conversion is not the answer to this problem, since UC can't be used as the target in an assignment and copying is too inefficient.

RR-0417 Length clause should force allocation of EXACT number of bits
The interpretation of length clauses is actively under review by the ARG. In particular, see AI-00536, AI-00553, AI-00561, and AI-00825.

RR-0626 Files produced by SEQUENTIAL_IO and DIRECT_IO are not portable among compilers, even for the same target machine e.g., because of dope vectors

RR-0773 Need to pack variable-length records into a block for data transmission

6.3 Interrupts

User Need U6.3-A: Interrupt Handling

Requirement R6.3-A(1) — Interrupt Servicing

Several user requests relevant to this section noted that Ada was over-restrictive in requiring that interrupt priorities be uniformly higher than user-defined task priorities. Others simply asked for more efficient and natural ways to deal with interrupts.

RR-0087 Allow software priorities to match/exceed hardware priorities

RR-0115 Provide better interrupt handling model
This RR contains a good discussion of current problems in dealing with interrupts.

RR-0151 Need standard support for priority interrupts

RR-0179 The treatment of interrupts is too implementation-dependent
Several problems are discussed in detail in this RR.

RR-0286D Interrupts should be handled with a procedure model, not a task model

RR-0316 Improve interrupt handling, e.g., with interrupt procedures

RR-0421A Need to delay in processing an interrupt

RR-0421D The treatment of interrupts as ordinary, timed, or conditional calls may depend inappropriately on the run-time system
The point here is that the run-time system may insulate the application program too completely from hardware-dependent behavior, and so different implementations may behave differently even for the same target hardware.

RR-0686 Priority of interrupts higher than normal tasks is ill-conceived

Requirement R6.3-A(2) — Interrupt Binding

Revision requests relevant to this section dealt with various aspects of Ada's model for associating interrupts with code to be executed when an interrupt occurs.

RR-0114 Allow an address clause for each task instance, and not just on the type
Meeting this requirement should solve the problem underlying this RR.

RR-0195 Need interrupt address per task, not task type
Meeting this requirement should solve the problem underlying this RR.

RR-0349 Interrupt addresses and memory addresses are conceptually different and should not be treated the same by the language
This RR presents what is believed to be a potential problem, but does not give any specific example of a difficulty imposed by the current approach.

RR-0421B Interrupt address structure is sometimes different from memory address structure; a single type for both is inappropriate
No specific examples of problems are given.

RR-0421C Need to associate interrupts with entries of task objects, not task types

RR-0710 Need to tie task entries to asynchronous external events generated by operating system

RR-0735 Need ability to change interrupt bindings at run-time

6.4 Dynamic References to Global Objects

User Need U6.4-A: Dynamic Access to Global Objects

Requirement R6.4-A(1) — Access Values Designating Global Objects

The concern here was primarily the ability to get access to special memory locations and to data structures that may have been initialized outside the program.

RR-0018 Need pre-elaborated constant arrays with variable-sized elements
This RR gives a careful and extensive discussion of the stated need.

RR-0110 Provide explicit control over placement of and access to data in different types or regions of memory

RR-0238 Allow access values to designate read-only memory

RR-0258 Need access values that point to declared objects
The purpose behind this request is to be able to establish static data structures linked by pointers.

RR-0291 Clarify whether use of an address clause causes storage to be initialized

RR-0293 Allow access values to point to declared objects
No examples are given.

RR-0338 Provide pointers to static objects and safe conversion between ADDRESS values and access values
Examples include large data structures such as maps residing in ROM. The use of unchecked conversion is too implementation-dependent and unsafe because addresses and access values do not necessarily have the same representation.

RR-0524 Allow functions to return references to components of objects; allow programmer to ensure pass by reference for any object

RR-0726 Need non-contiguous arrays, static pointers
The requirement supplies much of the requested functionality.

AI-00874 Ensure that access values are values of 'ADDRESS

User Need U6.4-B: Low-Level Manipulation of Access Values

Study Topic S6.4-B(1) — Low-Level Pointer Operations

One RR gave a specific example of the reason these capabilities are needed.

RR-0450 Need efficient manipulation of buffers whose type is determined at run time
The RR gives an example of a use of the capabilities called for in the requirement.

6: Requirements for System Programming

7 Requirements for Parallel Processing

7.1 Shared Memory

User Need U7.1-A: Control of Shared Memory

Requirement R7.1-A(1) — Control of Shared Memory

Deficiencies in Ada's treatment of shared memory were mentioned by several revision requests.

RR-0062 Ensure memory mapped devices are treated correctly by compilers

RR-0119 Need synchronized reference to elements of shared composite objects
This RR provides a good discussion of some problems concerning the use of memory locations shared among tasks, e.g., memory-mapped I/O and guarding against optimizations that remove references to volatile memory locations.

RR-0434 Need atomic read/write operations on shared volatile memory

RR-0678 Pragma SHARED is not sufficient for data shared between programs; need VOLATILE

AI-00142 Allow pragma SHARED to be applied to components of composite objects

7.2 Massively Parallel Architectures

User Need U7.2-A: Large Numbers of Tasks

Study Topic S7.2-A(1) — Managing Large Numbers of Tasks

The revision requests primarily address the need to initialize tasks and to give tasks unique identities.

RR-0123 Provide initialization values to tasks at startup
This RR provides an extensive discussion and examples illustrating the problem and a possible solution.

RR-0133 Allow a task component of an array to get its index
This RR explicitly cites an example for a massively parallel architecture.

RR-0334 Need to specify task parameters giving a task its work domain, e.g., to process part of an array

RR-0380 Need a task identifier for every task
The RR gives a lengthy discussion of possible uses of task identifiers.

7.3 Vector Architectures

User Need U7.3-A: Support for Vector Architectures

Study Topic S7.3-A(1) — Statement Level Parallelism

Revision requests for these capabilities come from users who are interested in computationally intensive numerical analysis.

RR-0514 Provide support for simple parallel threads within a program unit
Some of the requested functionality is included in the requirement.

RR-0738 Add facilities to support vector processing hardware
Although the requirement does not suggest that vector types and operands be added to the language, it does require that the revision address the needs of vector processing hardware.

RR-0741 Need hot performance on vector machines; add vector types and operands

7.4 Configuration of Parallel Programs

User Need U7.4-A: Configuration of Parallel Programs

Study Topic S7.4-A(1) — Configuration of Parallel Programs

There were no revision requests addressing this study topic, which arose from discussions with experts in the use of massively parallel systems. The ability to control the placement of tasks is, however, of interest in distributed systems as well (see Requirement R8.2-A(1)).

8 Requirements for Distributed Processing

8.1 Distribution of Ada Applications

User Need U8.1-A: Distributing an Application

Requirement R8.1-A(1) — Facilitating Software Distribution

This requirement deals with adjusting Ada semantics to allow for the possibility of distributed execution. It is stated as a Requirement rather than as a Study Topic because it is expected that only small changes are needed and because proposals for standard interprogram communication packages already exist.

RR-0109 Provide Ada semantics that are helpful when dealing with a single distributed Ada program
The request for distribution across heterogeneous processors is not met. This RR, however, gives a good discussion of some of the key problems that make use of Ada 83 more difficult than necessary for distributed processing.

RR-0111 Provide explicit support for fault tolerance and recovery
This is covered by item 2 of the Requirement.

RR-0182 Define visibility limits for parts of a program running on different processors

RR-0728 Need simple Ada run-time system for distributed memory MIMD architectures
This RR asks for simplifications that reduce the size of the runtime system that must be supported on each node of a distributed system. No specific suggestions are made.

There were several requests for a standard interprogram communication package. Such a need is mentioned in the discussion of the requirement.

RR-0181 Need standard means of communicating between Ada programs

RR-0222 Need additional predefined packages for process control/communication

RR-0224 Add communication support required for distributed systems

RR-0378 Need standard means of communication in distributed system

RR-0480 Need standard means of sending messages between Ada programs

8.2 Dynamic Reconfiguration of Distributed Systems

User Need U8.2-A: Configuring an Ada Application

Requirement R8.2-A(1) — Dynamic Reconfiguration

This requirement is not classified as a Study Topic because it was felt that only small changes are needed to allow for the possibility of static or dynamic configuration of programs. In particular, ensuring that elaboration of declarations takes place only when necessary is an important part of this requirement. Such control is also useful for non-distributed applications.

RR-0370A Can’t recover space declared in library units when reconfiguring a system

RR-0370B Can’t restart library level tasks

RR-0373 Need to be able to dynamically alter a program as it is running

RR-0377 Ada should allow partitioning of programs for multiple processor environments

RR-0661 Need language features for assigning tasks to nodes

RR-0665B Support allocation of parallel processes to processors

RR-0723 Need support for reconfiguration in emergency cases

8: Requirements for Distributed Processing

The following requests concern pre-elaboration; they also are met by Requirement R8.2-A(1) since the ability to avoid elaboration is essential during reconfiguration.

RR-0117 Provide pre-elaboratable constructs
This RR presents a set of possible rules defining units that can be pre-elaborated.

RR-0243 Allow/require elaboration prior to run time

RR-0244A Require pre-elaboration of some constructs

RR-0245 Change Standard to encourage pre-elaboration

RR-0246 Ensure that constant declarations are not elaborated at run time when initialized with static expressions
The problem addressed here is pre-elaboration, although the proposed solution is too drastic.

RR-0285 Minimize the need for run-time elaboration

RR-0639 Need compile-time initialization of complex data structures
This problem may be better solved with a separate CASE tool.

RR-0653 Need to declare constants whose value is supplied after linking

9 Requirements for Safety-Critical and Trusted Applications

9.1 Predictability of Execution

User Need U9.1-A: Constraining the Possible Meanings of a Program

Study Topic S9.1-A(1) — Determining Implementation Choices

This Study Topic asks that when freedom is given to implementations, the implementation should provide a way of controlling or at least document the choices that have been made. This Study Topic complements Requirement R2.4-A(1), which deals with reducing implementation dependences.

RR-0143 Document implementation dependences

RR-0176 Document run-time system performance and memory allocation strategies

RR-0644 Standard should specify time bounds/constraints for certain operations

This RR provides some possibly helpful examples of performance constraints that might be imposed in an Ada 9X annex.

Requirement R9.1-A(2) — Ensuring Canonical Application of Operations

The ability to ensure a compiler does not remove seemingly redundant checks is important in safety-critical applications.

RR-0254 Too much freedom is allowed with respect to exceptions and intermediate expression results

RR-0386 Need standard way of telling the compiler not to optimize

RR-0554 Need constraint checks for target of Unchecked_Conversion and I/O input

Section 11.6 of Standard allows seemingly redundant constraint checks to be optimized away.

RR-0718 Need predictable results in numeric computation, especially regarding optimization

This RR gives an example of how optimization might cause difficulty in evaluating carefully constructed numerical expressions.

RR-0729 Language should provide way to turn off optimization to eliminate bugs

9.2 Certifiability

User Need U9.2-A: Validation of Generated Code

Requirement R9.2-A(1) — Generating Easily Checked Code

This requirement was developed as a result of meetings with experts in the safety-critical and trusted systems community. No revision requests were submitted that are directly relevant to the requirement.

9.3 Enforcement of Safety-Critical Programming Practices

User Need U9.3-A: Restricting the Use of Ada Features

Requirement R9.3-A(1) — Allow Additional Compile-Time Restrictions

Solutions to this requirement are of general utility, but since the requirement is particularly important to safety-critical and trusted systems, it was placed in this chapter. The revision requests

suggest various restrictions that might be allowed by a capability proposed in response to the stated requirement.

RR-0216 Require that each task entry have at least one accept statement
Requiring at least one accept statement for each entry may be a reasonable project coding convention that should be enforceable by compilers.

RR-0325A Allow implementations to enforce local coding standards

RR-0328 Require compilers to report questionable uses of the language
This RR does not list any specific questionable uses.

RR-0435 Need secondary standard for simple Ada subset for safety-critical applications
The requirement does not propose that Ada 9X will provide such a standard, but it does allow an independently-developed standard to be enforced.

RR-0517 Provide syntax to declare program units free from side-effects
It is not clear that the benefits are worth the costs in language complexity and compiler checks.

RR-0538 Create new loop structure which bans the EXIT statement
A pragma could be used to forbid use of the exit statement.

RR-0771 Require tasks to have an accept for each entry
Requiring at least one accept statement for each entry may be a reasonable project coding convention that should be enforceable by compilers.

10 Requirements for Information Systems

10.1 Handling Currency Quantities for Information Systems

User Need U10.1-A: Support for Currency Quantities

Requirement R10.1-A(1) — Decimal-Based Types

This requirement was based on input from the information systems community in meetings with the Requirements Team.

Study Topic S10.1-A(2) — Specification of Decimal Representation

There was only one revision request directly related to the requirement.

RR-0357 Need packed decimal, wide-ranging fixed-point, decimal deltas

10.2 Compatibility with Other Character Sets

User Need U10.2-A: Alternate Character Set Support

Study Topic S10.2-A(1) — Alternate Character Set Support

AI-00216 Provide standard methods for testing whether characters are numeric, upper case, lower case, control, etc., independent of character representation

This AI requests that such tests be specifiable in a uniform manner, regardless of the representation for a character set.

10.3 Interfacing with Data Base Systems

User Need U10.3-A: Interfacing Ada Programs to DBMSs

Study Topic S10.3-A(1) — Interfacing with Data Base Systems

Interaction with DBMS's is increasingly common in building information systems. The requirement reflects this trend.

10.4 Common Functions

User Need U10.4-A: Standard Data Manipulation

Study Topic S10.4-A(1) — Varying-Length String Package

There were several revision requests asking for standard facilities in support of varying length strings.

RR-0051B Provide standard string manipulation packages

RR-0163 Need support for variable-length strings with appropriate equality and assignment operations

RR-0310 Need convenient way to pad with blanks in string assignments

A varying-length string library might obviate the need for this functionality.

RR-0327 Add varying length strings to the language

RR-0419 Add some form of support for varying length strings to the language

Study Topic S10.4-A(2) — String Manipulation Functions

String editing functionality was requested several times.

RR-0051C Provide packages for string edit functions

RR-0324 Add more flexible support for string manipulation
The RR suggests incorporating string manipulation operations that are supported in ICON, PL/I, and REXX.

RR-0360 Add picture-formatting capabilities to TEXT_IO
The requirement does not go this far, but does suggest adding picture-formatting functions in a separate package.

11 Requirements for Scientific and Mathematical Applications

11.1 Floating Point

User Need U11.1-A: Common Mathematical Functions

Requirement R11.1-A(1) — Standard Mathematics Packages

Two standard packages are currently under development by numeric working groups. One package deals with primitive functions useful in developing numeric algorithms; the other specifies standard elementary functions. Both packages could be included in Ada 9X either directly or by reference. Two revision requests asked for functionality that is not included in either of these packages.

RR-0308 Add libraries for array processing

RR-0536 Provide MIN and MAX numeric operators

The following requests ask for primitive functions useful in developing numeric algorithms. These functions are being provided in the Generic Primitive Functions package currently being considered for ISO standardization.

RR-0024 Need a way to decompose floating point numbers into mantissa/exponent

RR-0102 Provide explicit remainder operator for real numbers

RR-0255 Provide a function for returning the value of the next floating point number

RR-0346 Need portable way to extract mantissa/exponent from floating point number

RR-0358 Need support for floor, ceiling, truncate, and whole operations

RR-0453 Provide a special function or attribute yielding the sign of a numeric value

RR-0454 Need Entier function or attribute for real types

RR-0535 Provide CEILING and FLOOR numeric operators

RR-0645 Need mantissa/exponent extraction and manipulation

RR-0716 Unify and add attributes for numeric types

Many of the requested functions are being provided in packages currently being considered for ISO standardization.

These requests ask for standard functions such as trigonometric functions, logarithms, square root, etc. These functions are being provided in the Generic Elementary Functions package currently being considered for ISO standardization.

RR-0051A Provide common mathematics packages

RR-0189 Standard should include a floating-point math library interface

RR-0348 Need predefined functions for real numbers, e.g., trig, log, etc

RR-0719 Need standard for trig functions, sqrt, etc

User Need U11.1-B: Floating Point Support

Study Topic S11.1-B(1) — Floating Point Facilities

The requirement here is to foster predictable use of floating point on a variety of architectures. Given the differences in architectures, this implies providing a set of functions that allow algorithms to be adapted to the idiosyncrasies of particular architectures. The following revision requests reflect the difficulties in using the current floating point model predictably.

RR-0225 Ensure floating point representation with desired accuracy is used

RR-0252A Ensure support for IEEE floating point standard; allow full use of machine characteristics

RR-0252B Programmer needs to know/control whether rounding or truncation is used in real calculations

RR-0252C Ensure programmer can choose appropriate floating point representation

RR-0252E Provide a floating point model that reflects actual machine architecture

RR-0369 Provide support for floating point standard IEEE-754

RR-0492 Decouple mantissa and exponent information in floating point type definitions

RR-0564 Allow implementation freedom to include more mantissa digits in floating point safe numbers

RR-0636 Improve Ada's axioms for floating point operations

RR-0637 Ada programs should run as though negative zero did not exist

This is a complex issue that is being considered by the Numerics Rapporteur Group of ISO-IEC/JTC1/SC22/WG9.

RR-0720 Floating-point model should reflect actual hardware architectures

RR-0731 Use the Language Compatible Arithmetic Standard as a basis for Ada's floating point model

AI-00609 Floating point machine attributes inadequate to fully characterize machine characteristics

11.2 Representation of Arrays

User Need U11.2-A: Ordering of Array Components in Memory

Study Topic S11.2-A(1) — Array Representation

The RRs here are concerned primarily with interfacing with libraries of FORTRAN subroutines.

RR-0039 Make it easier to access FORTRAN libraries

This RR discusses quite extensively the problems of interfacing Ada with FORTRAN numerics subroutines.

RR-0507 Provide information/control over row-major or column-major ordering

The RR gives a detailed discussion of the inefficiency caused by Ada rules.

12 Efficiency, Simplicity, and Consistency Issues

This chapter corresponds to Appendix A of the Requirements Document. Section 2.2 of that document calls for general improvements to the language by reducing deterrents to efficiency, modifying rules that have proven to be confusing or error-prone to users, and minimizing special case restrictions.

Only some of the RRs relevant to the requirements in Section 2.2 were reflected in Appendix A of the Requirements Document. Additional topics that should be considered by the Mapping/ Revision Team are listed in Section 2.2 above.

12.1 Efficiency of Executed Code

12.1.1 Access to a Task Outside its Master (see RD A.1.1)

These RRs point out the cost in execution efficiency caused by allowing a task to be accessed from outside of its master.

RR-0104 Prohibit access to a task outside its master

RR-0194 Disallow referencing a task from outside its master

RR-0427 Do not permit a function to return a locally-declared task object

AI-00570 Releasing heap storage associated with task type instances
Implementations would find it easier to return unused storage if tasks could not exist outside their masters.

12.1.2 Null Ranges (see RD A.1.2)

These RRs point out problems with null ranges.

RR-0234 ‘‘Sub-null’’ ranges are of little value and an implementation burden

RR-0249 ’First and ’last for null ranges are defined oddly
This RR gives a specific example of a problem.

RR-0250 Define clearer notation for expressing null ranges

12.2 Understandability

Appendix Section A.2 of the Requirements Document lists some language rules that have proven to be confusing or error-prone to users and that therefore should be considered for revision.

12.2.1 Elaboration Order (see RD A.2.1)

These RRs point out problems with controlling elaboration order, particularly in large applications where programmers do not have access to all the code that must be elaborated.

RR-0004 Pragma ELABORATE should be transitive

RR-0218 Make the implementation find a good library-unit elaboration order
The problem is relevant to a revision of pragma ELABORATE.

RR-0233 Pragma ELABORATE should be transitive

RR-0396 Add library unit elaboration ordering rules to reduce need for pragma ELABORATE

RR-0546 It is too difficult to ensure that pragma ELABORATE is used when it is needed
This RR gives some examples of problems involving pragma ELABORATE.

RR-0581A Eliminate need for pragma ELABORATE; pragma NOT_ELABORATE might help
This RR contains some detailed discussion and examples.

RR-0581 Rules specifying the position of pragma ELABORATE are error-prone and unhelpful

RR-0767 Solve the elaboration order problem without requiring the use of pragma ELABORATE

AI-00421 Eliminate pragma ELABORATE
This AI explains the dangers in the current definition of pragma ELABORATE.

12.2.2 Later Declarative Items (see RD A.2.2)

These RRs point out the confusion that arises from the restriction that certain forms of declaration are not allowed after a body has been declared.

RR-0032 Allow grouping of variable declarations and related subprograms

RR-0428 Order of declarations is too restrictive
Specific anomalies mentioned are the inability to specify an address clause immediately after an entry declaration and the inability to specify a representation clause after a body has been declared.

RR-0569 Relax rules separating basic from later declarative items

RR-0594 Relax rules separating basic from later declarative items

12.2.3 Visibility of Literals and Operations (see RD A.2.3)

Quite a few revision requests asked that Ada 9X provide a way to make the operators of a type directly visible without writing a use clause.

RR-0022 Need direct visibility of operators declared in another package

RR-0057 Need direct visibility to infix operators in another package

RR-0096A Permit renaming an enumeration literal as a character literal

RR-0232 Need to allow direct visibility of operators in packages

RR-0239A Renaming an enumeration type should make literals visible

RR-0393 Can’t get direct visibility of fixed point mult and div operator by renaming

RR-0429 Need construct that makes just overloadable declarations directly visible

RR-0467 Need convenient way to rename a type and get its operations

RR-0474 Need direct visibility to just enumeration literals and operators of a type

RR-0555 Need ‘‘selective’’ USE clause to get just operators and subprograms of a type

RR-0624 Provide selective direct visibility into a package
The requirement addresses some of the request.

RR-0652 Declaring a subtype should make the equality operator directly visible

RR-0694 Need easy direct visibility to the equality operations

RR-0727 Need selective direct visibility of package declarations
The requirement reflects some of the requested functionality.

AI-00378 Enumeration literals should be made directly visible by a subtype declaration

AI-00390 Character literals should be made directly visible by a subtype declaration

AI-00480 Operators should be made directly visible by a subtype declaration

12.2.4 Obsolete Optional Bodies (see RD A.2.4)

The rules dealing with optional package bodies can cause subtle program errors.

RR-0426A The effect of an optional package body is confusing to users

RR-0689 Optional bodies should not be unlinked without a warning

12.2.5 OTHERS Choice in Aggregates (see RD A.2.5)

The rules restricting the use of an OTHERS choice have proven to be confusing to programmers.

RR-0029 Allow use of OTHERS with named associations when the index constraint is determined by context

RR-0571A Allow use of OTHERS choice with named associations when index bounds are determined by context

RR-0605 Rules for OTHERS in aggregates are confusing

12.3 Generality

Appendix Section A.3 of the Requirements Document lists some examples of existing capabilities that could be generalized in natural ways.

12.3.1 IMAGE and VALUE for Real Types (see RD A.3.1)

There were two requests for generalizing the use of the IMAGE and VALUE attributes.

RR-0363 Allow 'VALUE and 'IMAGE to apply to real types as well as discrete types

RR-0664 Need 'IMAGE and 'VALUE attributes for floating-point types

12.3.2 Exception Handlers in Accept Statements (see RD A.3.2)

One revision request pointed out this seeming inconsistency in the language.

RR-0499 Like other ''blocks'', allow exception handlers in accept statements

12.3.3 RANGE Attribute for Scalar Types (see RD A.3.3)

The RANGE attribute was not defined for scalar types because in many cases, the subtype name itself suffices. For example:

```
type ARR is array ARR_TYPE(SCALAR_SUBTYPE) of INTEGER;
```

It is not necessary to write:

```
type ARR is array ARR_TYPE(SCALAR_SUBTYPE'RANGE) of INTEGER;
```

Nonetheless, from a consistency viewpoint, users expect to be able to use the RANGE attribute in this situation.

RR-0155 Define RANGE attribute for scalar types

RR-0304 Define RANGE attribute for scalar types

RR-0623 Define RANGE attribute for discrete ranges

12.3.4 Permit "raise ... when <condition>" (see RD A.3.4)

For many programmers, it would seem consistent to be able to use a when clause on **return** and **raise** statements as well as on **exit** statements.

RR-0132 Allow optional WHEN <condition> on RAISE statement for consistency with EXIT statement

RR-0141 Allow WHEN <condition> on RAISE statements

RR-0200 Allow optional when_clause on RAISE and RETURN statements

RR-0362 Allow optional when_clause on the raise statement

RR-0614 Allow WHEN condition RETURN to make selection of returned value clearer

RR-0751 Add WHEN/RAISE construct to the language

12.3.5 STORAGE_SIZE for Task Objects (see RD A.3.5)

Several requests noted the need to specify the storage size for individual task objects rather than just on task types.

RR-0464 Should be able to set STORAGE_SIZE for task objects as well as types

RR-0648 Need to set STORAGE_SIZE on task objects, not task types

RR-0703 Need to specify STORAGE_SIZE on task objects, not task types

AI-00453 STORAGE_SIZE for tasks

12.3.6 Explicit Type Conversions in Static Expressions (see RD A.3.6)

An Ada comment, which led to the current rule, is reproduced in Appendix C.

RR-0009 Allow static conversion to static discrete type of static discrete expression

It might be worthwhile to go even further to allow general static scalar conversions, despite the problem of INTEGER(1.5). See Ada Comment 4709, which is reproduced here as Appendix C.1 on page 102.

RR-0099 Explicit type conversions should be allowed in static expressions

The RR gives an example of a problem caused by the current rules.

12.3.7 Use of a Subprogram Name in its Specification (see RD A.3.7)

RR-0675 points out the seeming inconsistency in the following example that stems from the restriction in section 8.3(16).

```
with TEXT_IO;
package USER is
    function MODE(FILE : TEXT_IO.FILE_TYPE) return TEXT_IO.MODE; -- illegal
    function NAME(FILE : TEXT_IO.FILE_TYPE) return STRING
        renames TEXT_IO.NAME; -- legal
end USER;
```

RR-0462 Allow selected component form of type mark in a formal part even when the selected component has the same identifier as the subprogram

RR-0483 Allow an instantiated subprogram to have the same identifier as the generic unit (as is allowed for package instances)

RR-0579 Allow a type mark of form P.FOO in the formal part of a subprogram named FOO

RR-0675 Allow a subprogram identifier to be used as a type mark in its specification

12.3.8 Default Names for Generic Formal Parameters (see RD A.3.8)

The following RR provides a good example of the problem.

RR-0714 Allow default names for all generic formal parameters
The RR gives a detailed example showing the problem caused by the inability to associate default names with generic formal types.

12.3.9 Ability to Redefine "=" (see RD A.3.9)

Several RRs noted that it would be useful to allow the straightforward declaration of the equality operator for all types.

RR-0008 Allow overloading of the equality operator for all types

RR-0025 Allow overloading of the equality operator with different operand types

RR-0412 Allow overloaded = for all types, not just limited types

RR-0513 Allow overloading of = for any type, e.g., returning an array type
As the RR points out, there is no strong reason to limit the result type of the equality operators.

12.3.10 Reading OUT Parameters (see RD A.3.10)

RR-0002 points out a problem with the current restriction on reading **out** parameters — since programmers assign results to a local surrogate for the **out** parameter, they can easily forget the final assignment from the surrogate to the formal **out** parameter before returning from a procedure.

RR-0002 Allow reading of OUT parameters

RR-0303 Allow reading of OUT parameters

RR-0539 Allow reading of OUT parameters

AI-00478 Allow reading of OUT formal parameters
This AI points out that if a formal **out** parameter is used as an actual **out** parameter in a call, it is quite natural to want to read the returned value before returning from a call.

AI-00479 Initialize access type OUT parameters to null
This AI is essentially the same as RR-0559.

12.3.11 Implicit Subtype Conversions (see RD A.3.11)

Many programmers are caught by surprise when no implicit subtype conversion is allowed for array aggregates in certain contexts.

RR-0240 Non-sliding aggregates and slices in component associations
The RR points out inconsistencies between assignment and component association.

RR-0573 Slide indices of array aggregates for record component initialization and as components of record aggregates

RR-0734 Generalize cases that allow implicit subtype conversion

RR-0749 Should allow index sliding for slices serving as actual parameters and as values in record components

12.3.12 Negative Literals in Loops (see RD A.3.12)

The Ada comment that led to the current rule is reproduced in Appendix C.2 on page 103.

RR-0156 A negative literal should be allowed wherever a literal is allowed

AI-00140 Allow -1..10 as a discrete range in loops

12.3.13 Naming Syntactic Items (see RD A.3.13)

Several RRs suggested that the ability to name constructs should be extended to more than just blocks and program units.

RR-0199 Allow IF, CASE, and SELECT constructs to be named

RR-0205 Allow program unit name on PRIVATE, BEGIN, and EXCEPTION

RR-0340 Allow optional simple name on CASE, IF, and SELECT statements

RR-0596 Allow END type_name to substitute for END RECORD

RR-0673 Allow ‘‘END RECORD type_name’’ to substitute for ‘‘END RECORD’’)

12.4 Usability of Ada

Appendix Section A.4 of the Requirements Document lists some areas in which changes to Ada will make it easier to use.

12.4.1 Completion of Subprogram Declarations (see RD A.4.1)

The ability to complete subprogram declarations with a renaming declaration or a generic instantiation was often requested.

RR-0055 Allow a subprogram body to be defined by renaming or generic instantiation

RR-0096B Allow a procedure body to be provided by a renaming declaration

RR-0157 Allow renaming when defining a subprogram body
This RR gives examples showing the usefulness of the proposed capability.

RR-0231 Allow a rename definition of a subprogram body
Examples of inadequate workarounds are given.

RR-0364 Allow a subprogram body to be defined by generic instantiation
An example is given using an instantiation of UNCHECKED_CONVERSION.

RR-0470 Allow renaming or generic instantiation to define a subprogram body
Reasonable examples are given.

RR-0550 Allow subprogram bodies to be defined by RENAMES or generic instantiation

RR-0666 Allow a subprogram body to be given by generic instantiation

RR-0667 Allow a subprogram body to be given by RENAMES

RR-0725 Need rename in package body for routine in package specification

RR-0764 Allow subprogram bodies to be defined by RENAMES
This RR argues that the workaround needed when a subprogram body can’t be provided by a renaming declaration increases recompilation requirements.

12.4.2 Completing Incomplete and Private Types by Subtype Declarations (see RD A.4.2)

The ability to use a subtype declaration to provide the full declaration of a private type was proposed in several revision requests.

RR-0096C Allow the full declaration of a private type to be provided by a renaming declaration

RR-0690 Allow incomplete and private types to be completed by subtype declaration

AI-00540 Completing a private type declaration with a subtype declaration

13 Requests Rejected

Rejected requests are listed in this chapter. Except for the first set of RRs, they are grouped by reason for rejection.

13.1 Requests Rejected: Various Reasons

Each request listed in this section is accompanied by a short description of its reason for rejection.

RR-0146 Support for file/record locking
This is too specialized a capability to require for every implementation.

RR-0147 Add support for ISAM
Although an ISAM package might be useful, there are too many other higher priority requirements that should be addressed.

RR-0153 Private part foils separation of specification and implementation
Much of the requested functionality can be obtained by completing an incomplete type in a package body.

RR-0154 Subunits should not have to be at the outermost compilation unit level
The ability to declare a subunit in a nested block would require extra complications in requiring that all enclosing blocks be named. Allowing subunit declarations in nested units but not in blocks would seem to be a non-uniformity, so there is no easy way to provide the requested capability.

RR-0162 Provide a clean interface to a SORT package
Providing attributes for use with a standard interface to a sort package would be useful in information system applications, but other changes were judged to have higher priority.

RR-0214 Require that a subprogram parameter be used within the body
Such a change would be an inconvenience during program development.

RR-0217 Require that a parameter of an entry be used within an accept
Such a change would be an inconvenience during program development.

RR-0248 Allow users to specify locations for discriminants that are outside record values
The RR does not provide sufficient justification for allowing non-local record discriminants.

RR-0253 DIGITS and DELTA approach leads to inefficiency, non-portability
This RR does not reflect a correct understanding of the efficiency impacts of DIGITS and DELTA specifications.

RR-0256 Fixed-point approach with range and delta is not what is needed
Fixed point representations can be completely controlled in Ada 83 with proper use of 'SMALL and 'SIZE representation clauses.

RR-0263 CONSTRAINT_ERROR is too broadly defined
This issue was given thorough consideration in the original design. Insufficient evidence is given in this RR to justify reconsidering the decision.

RR-0277 Inappropriate wording
The wording (in 8.6(1), not 8.8(1) as in the RR) is acceptable. For 9(5) the comment refers to a note, which is worded acceptably.

RR-0299 Make everything in the Standard ''part of the standard''
Many readers find the extra material useful.

RR-0322 Do not add any new reserved words to the language
This matter will be resolved by the Mapping/Revision Team, and there is no direction in the Requirements Document. Note, however, that the magnitude of the requirements is such that it is not likely to be practical to meet this request. See also the Upward Compatibility guideline on page 5 of the Requirements Document.

13: Requests Rejected

RR-0329 Using a deferred constant before it has a value
The apparent problem raised by this request does not exist. The example given in the RR is illegal by 7.4.1(3).

RR-0339 Support sorting in extended alphabets
There does not appear to be any solution at the language level. See the discussion following Requirement R3.1-A(1).

RR-0345 Need standardized interface to other ANSI languages
Since interfaces to other programming languages depend on both the language and the implementation, it isn't clear that anything useful can be done to solve the RR's problem in Ada 9X, despite the example solutions given in the RR.

RR-0399 Break up overly broad predefined exceptions, e.g., CONSTRAINT_ERROR
This issue was given thorough consideration in the original design, and insufficient evidence is given in this RR to justify reconsidering the decision.

RR-0425 Need open ranges in declarations of real subtypes
There is no obvious notation, and the change is of marginal benefit.

RR-0449 Do not allow unchecked conversion of private types
Unchecked conversion exists as an escape mechanism whose usage should not be restricted by the language. Local controls on its use could be enforced in response to Requirement R9.3-A(1).

RR-0478 Add language facilities for restricting use of resources to trusted packages
Providing special-purposes pragmas for such purposes is beyond the scope of the revision effort.

RR-0479 Need standard subprograms to get user-interface information from OS
This is not a bad idea, but there are more important issues that deserve attention.

RR-0518 Provide syntax to declare subprogram pre/post conditions
The desired checks can be written in the existing language in a way that permits the optimizer to take advantage of the checks.

RR-0519 Simplify overload rules for ambiguous/universal expressions
These issues were considered thoroughly in the original design, and it is unlikely that the rules can be improved in general without introducing other anomalies. Of course, the -1..10 case (Section A.3.12) should be fixed, but this is not an overloading resolution anomaly but rather a special case rule.

RR-0520 Language should distinguish ''sequence'' and ''mapping'' arrays
It is far from clear that adding a new type would create a simpler, less easily misused language.

RR-0545 Subunits should not have to be at the outermost compilation unit level
The ability to declare a subunit in a nested block would require extra complications in requiring that all enclosing blocks be named. Allowing subunit declarations in nested units but not in blocks would seem to be a non-uniformity, so there is no easy way to provide the requested capability.

RR-0570 Allow the prefix of a name to denote a renaming of an enclosing construct
AI-00119 discusses the reasons for this restriction.

RR-0572 Need predefined operators with respect to all predefined integer types
The change would require revision of the overloading rules because X**2 would become ambiguous.

RR-0607 Allow names of compilation units to be overloadable, operator symbols
Although it may seem more uniform to allow library unit names to be overloaded, a with clause naming such a unit would be unresolvably ambiguous.

RR-0618 Ban GOTO statement
Tools that produce Ada code need to be able to generate GOTOs.

RR-0638 Axioms for built-in operations should be specified explicitly
This was considered and rejected in the initial design as being unnecessary for clarity and precision. The current wording is adequate.

RR-0680 Predefined exponentiation should take any integer type for exponent
This change is difficult to make because of the overload resolution rules. This problem was considered extensively in the initial design, and all solutions posed difficulties to users. There are more important changes to focus on in this revision of Ada.

RR-0700 Ensure that constant functions like sin(10.0) are evaluated at compile-time
It is too difficult to define what functions should be evaluated at compile-time. Moreover, the change would pose the potentially severe implementation burden of requiring a target machine function to be evaluated in the host machine environment.

RR-0765 Allow ''when Package_Name.others =>'' as exception handler
This change could introduce serious problems during maintenance.

RR-0774D Allow overloaded names in the library
Although it may seem more uniform to allow library unit names to be overloaded, a with clause naming such a unit would be unresolvably ambiguous.

RR-0774I Create separate standards, such as X-Windows, SQL
The creation of separate standards is outside the scope of the Ada 9X revision effort.

RR-0774L Allow pragma INTERFACE within a package body
The compiler needs this information before a package body is compiled in order to minimize the need for recompilation.

RR-0774M Allow a subprogram to be renamed in a body
Since a renaming declaration already is allowed in a body, the intent behind this request is unclear.

AI-00003 Allow data of mode IN in SEND_CONTROL
There is no requirement to fix the low-level I/O programming capabilities in the language. Other needs are more important.

AI-00274 Proposed extension of the USE clause — record component visibility
Introducing a Pascal-like **use** clause for records might be convenient, but it is not necessarily straightforward to ensure that all components of the record maintain their existence throughout the scope of the **use** clause. There are more important requirements to be addressed.

AI-00460 Allow non-integral powers for exponentiation
This change is difficult to make because of the overload resolution rules. This problem was considered extensively in the initial design, and all solutions posed difficulties to users. There are more important changes to focus on in this revision of Ada.

AI-00529 Resolving the meaning of an attribute name
The rules for resolving the overloading of an attribute prefix were adopted after considerable review of complex cases. The example given in this AI does not suggest that there is sufficient user need to reconsider this complicated area of the language.

13.1.1 Requests Rejected: Machine Code

These requests are for improvements in the machine code feature of Ada. They have been rejected because it is not considered worth the effort to make improvements to machine code procedures. Programmers can alternatively use direct interfaces to routines coded in assembly language.

RR-0043 Make it easier and more portable to use assembler with Ada

RR-0284 Machine-code insertions are unreadable; replace with INLINE macros

RR-0371 Need more usable and portable machine code insertions

RR-0489 Allow machine-code insertions in functions as well as procedures

RR-0691 Allow machine-code insertions in functions as well as procedures

13.1.2 Requests Rejected: Exit from a Block

Although there were several requests for extending the permitted use of exit statements to allow an exit from a block, there are difficult language design problems associated with such an extension. In particular, when a block is nested in a loop, there is no convenient syntax to distinguish whether an exit statement is intended to exit from the block or from the loop.

RR-0491 Code would be clearer if one could EXIT from a block statement

RR-0632 Allow EXIT from a block statement for consistency

RR-0695 Allow EXIT from block for legibility

13.2 Requests Rejected: An Opposing Requirement was Imposed

These requests were rejected because the Requirements Document states a contradictory requirement for reasons stated in the document.

RR-0044 There is no need to add unsigned integers to Ada
See User Need U6.1-A.

RR-0054 Do not add variable length strings to the language
See Study Topic S10.4-A(1).

RR-0071 Improve support for heterogeneous distributed processing
Dealing with heterogeneous distributed systems is beyond the scope of the revision (see Requirement R8.1-A(1)), but this RR discusses some of the issues that would have to be addressed otherwise.

RR-0300 Use an LR grammar to define the syntax of the language
Requirement R2.1-B(1) discourages such changes.

RR-0326 Use a different syntax production style
This RR suggests that the Ada syntax productions should provide more information about program legality and suggests that an attribute grammar should be used. This kind of stylistic change has been ruled out of scope by Requirement R2.1-B(1).

RR-0372 Solve problem where heterogeneous processors view memory differently
Dealing with heterogeneous shared memory systems is beyond the scope of the requirements (see Requirement R8.1-A(1)).

RR-0630 Due to high implementation costs, define/allow Ada subsets
The Requirements Team explicitly considered and rejected the notion of allowing subsets as not being consistent with the goals of the revision effort. See Section 1.2 of the Requirements Document.

RR-0646 Allow exceptions to be parameterized with parameters read in handler
The requirements for exceptions are much less ambitious because of potential implementation overhead.

13.3 Requests Rejected: Insufficient Information in the Request

The text of these requests provided insufficient information to permit the Requirements Team to evaluate them properly. In many cases the request states that a problem exists without explaining just what the problem is and what needs to be done about it.

RR-0012 Mutation of types is needed for AI applications
It isn't clear what capability is being requested.

RR-0080 Derived types are clumsy
The RR states that there are problems but does not identify them.

RR-0144 Require support for fixed point arithmetic even if floating point hardware is not present
It is not clear what language change, if any, is being requested.

RR-0158 Allow multi-way conditional and timed entry calls

RR-0166 Allow definition of the literal representations of an abstract data type

RR-0173 Allow a rendezvous with a higher-level entity, i.e., a set of tasks

RR-0186 It is difficult to write an entire operating system in Ada
The request asks for additional ways to refer to a task and to control it, but gives no examples of what deficiencies are to be remedied or what additional control is thought to be needed.

RR-0264 Discriminants need to stand out more

RR-0282 Ada program structure hides important context information

RR-0333 More precise definition of TEXT_IO is needed, less implementation freedom
The RR says that there are problems but does not identify them.

RR-0376 Need special treatment of exceptions in distributed/parallel/multi-processor systems
The RR is very short and does not clearly indicate what problem needs to be solved.

RR-0394 Merge concepts of task and package into concept of an object
Insufficient motivation is given for the requested change.

RR-0530 Insufficient support for mutants of limited types

RR-0608 Allow recursive generic instantiations
It is not clear how this could be implemented.

RR-0612 Should allow both delay and terminate alternatives in selective wait
The intended semantics is not clear.

RR-0699 Do not treat an unaccepted length clause for a type as an error

RR-0704 Make every bit available to the application programmer

RR-0759 Add real-time and verification facilities for control engineering

AI-00521 Fixed point subtypes should not inherit SMALL
The submitted comment does not give enough motivation for the suggested change to understand why a change might be useful.

13.4 Requests Rejected: Insufficient User Benefit

For these requests, it was judged that the item provided insufficient user benefit to justify disturbing the language even though the idea itself may be reasonable — not all reasonable ideas have sufficient benefit to justify incorporation in the revised language.

RR-0028 Add a semicolon terminator to SEPARATE statement syntax
Allowing an optional semicolon would not avoid confusion and requiring a semicolon would not be upward compatible or of much benefit.

RR-0049 Allow special notation when the same name is on both sides of :=

RR-0053 Allow aggregates for null records and arrays
The RR points out that a null array aggregate can't be written if the component type is a private type for which no values are directly available. Nonetheless, it doesn't seem worthwhile to invent new notation just for these special cases.

RR-0077 Provide stream I/O for digital signal processing
The need for stream I/O to support digital signal processing is too narrow for motivating a change, given the intended scope of the Ada 9X revision.

RR-0086 Need to initialize a record component to the address of the record itself

13: Requests Rejected

RR-0093 Allow full declaration of deferred constants to be given in a package body
Although the ability to defer the initialization of a constant to the body of a package would reduce the need for recompilation and although the RR proposes reasonable syntax, it is not clear that there is much demand for this change, and it would make new kinds of user errors possible (namely, accessing an uninitialized constant value).

RR-0095 Allow applicable units to be named in USE clauses and pragma ELABORATE

RR-0097 Allow/require explicit action to get default parameter value

RR-0098 Generalize incomplete typing for types other than access or private
The limitation mentioned here may be satisfied by changes made under Study Topic S4.3-B(1).

RR-0100 Allow constants to use default values to get value

RR-0126 Allow underscore before ''E'' in exponents

RR-0127 Allow real number output in non-decimal bases

RR-0131 In a qualified expression, should have visibility of the enumeration literals of the qualifying type

RR-0135 Catenation should not raise CONSTRAINT_ERROR for intermediate results

RR-0169 Allow ''null'' procedures for actual or default generic formal subprogram values
This problem is discussed in more detail in the Ada 9X report, ''Ada Support for Software Reuse'' [6].

RR-0198 Allow positional aggregate for single-component aggregate

RR-0208 Need ability to initiate TEXT_IO, DIRECT_IO, and SEQ_IO operations without waiting for completion
The need for high level asynchronous I/O is not sufficiently great to warrant a change to implementations.

RR-0221 Need to write common code for group of exception handlers

RR-0229 Need to hide the range of a scalar type and the initial value of an object to ensure these values are not used directly by programmers

RR-0270 Allow specification of read-only data from a package

RR-0297 LOW_LEVEL_IO was a bad idea; remove this package from the language

RR-0313 Allow deferred constants of arbitrary (i.e., non-private) types

RR-0321 Permit anonymous array and record declarations for record components

RR-0343 Provide better facilities for conditional compilation

RR-0356 Need a way to get the compilation date and time within a program
This problem can be solved with a suitable environment tool.

RR-0389 There is a need for ''cyclic'' discrete types in the language

RR-0391 Clumsy syntax for based numbers, especially in aggregates

RR-0426D Optional index in 'FIRST (and others) causes problems

RR-0444 Let the user limit the places where a given exception can be raised

RR-0452 Allow constant functions in static expressions (or overloadable constants)
The RR gives no examples showing why such functions are needed in contexts where the language requires static expressions (e.g., in the choice of a case statement).

RR-0463 'Size is unclear; perhaps need 'Spacing and 'Allocation

RR-0469 Parameter names for language-defined pragmas should be defined

RR-0504 Add an exchange operator

RR-0548 Allow convenient syntax for instantiating a nested generic unit

RR-0552 Need ''padded'' line input with truncation and pad-fill to 'LENGTH

RR-0558 Deriver of type should be able to hide subset of derived operations
The ability to hide some derived operations would add more complexity to the language than is acceptable to most users.

RR-0576 Allow parameter default expressions to make use of previous IN parameters
This request is harder to satisfy than may appear at first, since it requires, in effect, that actual parameters be evaluated in an order dictated by the order of the formal parameters and is a potential complication for implementers without strong benefit to users.

RR-0581C Allow a pragma ELABORATE for a subunit to mention a package name given in the context clause of a parent library unit

RR-0588 Provide a form of USE clause that hides outer homographs

RR-0621C Allow case statements to dispatch on value of an exception
The intent here is to call a procedure in an ‘‘others’’ handler with the actual exception as a parameter. A case statement in the procedure body dispatches on the actual exception value. This is an interesting idea, but not of sufficient value to fall under one of the stated requirements.

RR-0635 Provide basic support for extended precision integer arithmetic

RR-0642 Add label variables to support use of finite state machines

RR-0672 Need anonymous pointer types

RR-0679 Allow component selection on objects of a private type

RR-0696 Pragmas LIST and PAGE should be optional

RR-0701 Allow specification of STANDARD in the same way as for SYSTEM

RR-0730 The private part of a package should have its own context clause
Study Topic S4.3-B(1) may help to alleviate this problem.

AI-00211 Additional control statement to hop to end of the loop

AI-00442 Time zone information in package CALENDAR
It is not clear that time zone information is of sufficient general use to warrant a change to the language.

AI-00518 Fixed and floating type declarations needlessly different

AI-00538 Declaring constant arrays with an anonymous type

AI-00582 Need a standard name for null address

AI-00584 Restrict argument of RANGE attribute in Ada 9x
The example motivating the change is only one of several ways to write implementation-dependent programs and does not justify the proposed language change.

AI-00681 Can’t declare a constant of a NULL record type

13.4.1 User-Defined Attributes

These requests ask for the ability to declare user-defined attributes. They were rejected as not seeming to provide enough user benefit to justify disturbing the language.

RR-0406 Allow user-defined attributes for user-defined types

RR-0509 Allow user-defined attributes for user-defined or private types

RR-0613 User-defined attributes solve portability problems with implementation-defined attributes

RR-0674 Allow user-defined attributes as functions

13.4.2 Multidimensional Slices

These requests all concern multidimensional slices; they were rejected as not providing enough user benefit to justify disturbing the language.

RR-0323 Generalize slice for multidimensional arrays

RR-0494 Allow slices for any dimension in multidimensional arrays

RR-0508 Allow slices for any dimension in multidimensional arrays

13.4.3 Permit Functions to have Parameter Modes IN OUT and OUT

These requests are that function parameters not be restricted to having only mode **in**. This change is too great to be accepted by users, as evidenced by the comments received when it was proposed in a draft version of the Ada 9X Requirements.

RR-0026 Permit function parameters to have modes IN OUT and OUT

RR-0598 Permit function parameters to have modes OUT and IN OUT

13.4.4 Anonymous Arrays

Several requests asked for the ability to use anonymous array declarations more widely in the language, particularly as record components.

RR-0336 Allow array type definitions in records; nice for array-of-array case
LSN-222 discusses the potential complexity of allowing this capability. See Language Study Notes, 1983, available from the Ada Information Clearinghouse.

RR-0443 Need for anonymous array types as record components
LSN-222 discusses the potential complexity of allowing this capability. See Language Study Notes, 1983, available from the Ada Information Clearinghouse.

AI-00429 Allow array type definition for record component
LSN-222 discusses the potential complexity of allowing this capability. See Language Study Notes, 1983, available from the Ada Information Clearinghouse.

13.5 Requests Rejected: Too Much Implementor Change for the Payoff

These requests were rejected as requiring too much change by implementors to justify the expected payoff.

RR-0288 Integrate representation clause information with declarations

RR-0290 The syntax used in record representation clauses is hard to read

RR-0312 Generalize case statement to decision table

RR-0320 Generalize case statement for other types, including REAL

RR-0392 Need ‘‘semi-limited’’ type with predefined := but no predefined =

RR-0473 Allow ‘‘partially’’ constrained subtypes of discriminated records

RR-0529 Allow selection of operations based on run-time queries about properties of types
A fully general ability to query type descriptors at run-time is requested here.

RR-0708 Allow infix function calls

RR-0733 Need fixed-point types not centered on zero

13.5.1 User-Defined Operators

Two revision requests asked for syntactic extensions to Ada by allowing user-defined operator symbols. Such a change would pose parsing difficulties that outweigh the potential user benefits.

RR-0201A Liberalize overloading of operators to other character sequences

RR-0682 Allow user-defined overloaded operators such as ‘‘?’’, ‘‘:-’’, etc

13.5.2 Non-static Objects as Case Labels

These requests, for use of non-static objects as choices in case statements, were rejected as requiring too much change by implementors to justify the expected payoff.

RR-0561 Allow case statement to operate on strings for string processing

RR-0650 Allow non-static case statement choices, non-discrete case statement expression

AI-00477 Case choices should not have to be static

13.5.3 Discontiguous Subtypes

These requests ask for the ability to deal with discontiguous subtypes of enumeration types. This change was judged as requiring too much change by the implementors to justify the expected user benefit.

Several of these request a way to distinguish some elements of a discrete type from others. A constant array of booleans subscripted by the type provides an efficient solution to this problem.

RR-0031 Provide a way to test for a value in a non-contiguous set

RR-0046 Allow testing in discontiguous ranges and create true sets

RR-0058 Allow discontiguous subtypes of enumeration types

RR-0437 Provide ‘‘supertype’’ capability for merging enumeration types

RR-0603 Allow discontiguous subtypes of discrete types

13.5.4 Overload Names of Generic Units

These requests are for the ability to overload the names of generic units. This change was judged as requiring too much change by implementors to justify the expected user benefit.

RR-0035 Allow generic units to be overloaded

RR-0606 Allow generic subprogram names to be overloaded

13.6 Requests Rejected: Not Sufficiently Compatible with Ada 83

These requests were rejected because they either required too great a change from Ada 83 or the necessary change was not sufficiently upward compatible (in terms of Guideline **G-2**).

RR-0011 Expression 0**0 should not be 1 as this is an indeterminate form
No change could be upward compatible.

RR-0030 Require subprogram specification to list non-local objects referred to

RR-0079 TERMINATE alternative adds little value and is rarely used

RR-0134 Require re-evaluation of entry’count on abandoned entries
It is not clear that this would be easy or efficient to implement reliably due to race conditions. Once evaluation of a select statement has begun, no entry calls could be abandoned until one alternative has been selected.

RR-0152 Allow e.g., a < b < c which would mean a < b AND b < c
The RR proposes the new idea of n-ary operators.

RR-0197 For access types, parameter mode IN should mean the designated object cannot be modified

RR-0212 Allow assignment to record discriminant like other components

RR-0247 Don’t initialize access variables by default to NULL

RR-0251 Invent new notations to distinguish function call, array reference, and conversions

RR-0266 Operator overloading is dangerous

RR-0268 Separation of specification and body is not worth it

RR-0269 Make subprograms not recursive by default
This is a substantive change to the language, not just a change to a note as implied by the RR.

RR-0271 Distinguish storage classes for variables with key words like CONTROLLED or STATIC

RR-0272 Limited types are of little true value

RR-0273 There are problems with private types in the language

RR-0325B Allow implementations to experiment with supersets

RR-0366 Subtype natural should not include zero
This change would be dangerously non-upward compatible. Moreover, mathematicians disagree on this.

RR-0397 Replace keyword PRAGMA with something capturing meaning better

RR-0424 Allow names exported from an instance to be redefined during instantiation

RR-0426C Omitting index constraint in constant arrays causes programmer errors

RR-0471 Allow specification of parameter modes in subprogram calls for clarity
This feature was considered explicitly and rejected in the initial design.

RR-0476 Allow user-written type-conversion functions with the same name as the target type
The proposal would require significant change to the visibility rules and the overload resolution model.

RR-0495 Remove leading space in the result of the 'IMAGE attribute for integers

RR-0537 Separate integer divide and floating divide as in Pascal

RR-0559 If allow reading of OUT parameters, initialize OUT access to NULL
There is no particular advantage in initializing an **out** parameter to null rather than to the value of its actual parameter. Moreover, such a change would be inconsistent with the current rule for components of an access type.

RR-0617 Eliminate anonymous array types

RR-0620 Ban RETURN statement except inside functions

RR-0625 Change EXIT/WHEN to WHEN/EXIT to parallel Ada IF and English

RR-0670 Decouple = and /=; do not distinguish private from limited private

RR-0747 Provide better support for ''light-weight'' parallelism (as in Linda)
Although the LINDA model may be attractive, such a change of concept is outside the scope of the 9X revision effort.

RR-0753 Make syntax for task type declarations more consistent

RR-0774B Tasking defined as a standard package of functions

Two requests, concerned with providing dimensional mathematics support in Ada, were also judged as requiring too great a change from Ada 83.

RR-0354 Introduce dimensional mathematics into the language

RR-0745 Add facilities for dimensional mathematics to the language

13.7 Requests Rejected: Not a Language Issue

These requests are not properly addressed by changes to the language definition but rather by CASE tools or secondary standards.

RR-0150 Provide ''chaining'' of different programs to reduce memory requirements

RR-0210 Need more pragmas for software maintenance to MIL standards

RR-0259 Incomplete type declarations are dangerous and unnecessary
This request reflects an incorrect understanding of Ada.

RR-0262 Do not require existence of subunit for body stubs
A CASE tool can provide the requested functionality.

RR-0265 Allow implementations to short-circuit in general, forget AND THEN
This request reflects an incorrect understanding of Ada.

RR-0314 Define minimum-quality error diagnostics in the standard

RR-0351 Trusted systems require auto-scrubbing of memory when done with it

RR-0375 Include formal memory protection/security
The ability to restrict access to pages of memory is too operation-system dependent to be a suitable language requirement.

RR-0497 Presence of default discriminants for types used as generic actual can yield a surprising run-time error
The problem raised in this RR is really a problem of correctly using the language rather than a language problem, particularly since the proposed solution would allow a constrained access variable to inadvertently designate an incorrectly constrained object.

RR-0681 A definition of an Ada Line Of Code (LOC) should be standardized

RR-0746 Allow pictures/graphics as comments in source code
The environment should provide this functionality.

RR-0770 Make aborting yourself cause instant completeness
It is arguable that 9.10(6) requires immediate completion of a task that aborts itself.

AI-00427 Semi-constrained subtypes
This comment reflects a misunderstanding of the language.

13: Requests Rejected

References

1. *Ada 9X Project Requirements Workshop.* Office of the Under Secretary of Defense for Acquisition, Washington, D.C. 20301, June 1989.

2. *Ada Board's Recommended Ada 9X Strategy.* Office of the Under Secretary of Defense for Acquisition, Washington, D.C. 20301, 1988.

3. *Approved Ada Language Commentaries.* Published by Grebyn Corporation, Vienna, VA. See also the up-to-date files kept by the Ada Information Clearinghouse; these files are also available on the AJPO machine (ajpo.sei.cmu.edu).

4. *Reference Manual for the Ada Programming Language.* ANSI/MIL-STD-1815A-1983 edition, 1983.

5. Baker, T. "Timing Issues Working Group". *Ada Letters X*, 4 (Spring 1990), 119-135. Proceedings of the Third International Workshop on Real-Time Ada Issues.

6. Cohen, S. Ada Support for Software Reuse. Software Engineering Institute, October, 1990. Ada 9X Project Report.

7. Fowler, K. J. A Study of Implementation-Dependent Pragmas and Attributes in Ada. Software Engineering Institute, November, 1989. Ada 9X Project Report.

8. Quiggle, T. J. "Asynchronous Transfer of Control Working Group". *Ada Letters X*, 4 (Spring 1990), 15-24. Proceedings of the Third International Workshop on Real-Time Ada Issues.

9. *Ada 9X Requirements.* Office of the Under Secretary of Defense for Acquisition, Washington, D.C. 20301, 1990.

10. *Standard Generalized Markup Language (SGML).* ISO 8879 edition, 1987.

Appendix A: Numerical Listing of RRs

Revision request numbers are underlined when they contain examples or discussion that may be especially helpful to the Mapping/Revision Team or when reviewing proposed changes to the language.

RR-0001: *Limited types need assignment, constants.* This request is considered under Study Topic S4.2-A(2).

RR-0002: *Allow reading of OUT parameters.* This request is considered under Section A.3.10.

RR-0003: *Provide a compiler-independent finalization mechanism.* This request is considered under Study Topic S4.2-A(2). See the discussion following the Study Topic.

RR-0004: *Pragma ELABORATE should be transitive.* This request is considered under Section A.2.1.

RR-0005: *Exception declarations in generic packages make code sharing unnecessarily difficult.* This request is considered under Requirement R4.4-C(1).

RR-0006: *Distinguish unconstrained/constrained generic formal types.* This request is considered under Study Topic S4.4-B(2).

RR-0007: *Default representation for enumeration types should be specified.* This request is considered under Requirement R2.4-A(1). The representation for predefined type BOOLEAN should continue to be implementation-defined for efficiency reasons.

RR-0008: *Allow overloading of the equality operator for all types.* This request is considered under Section A.3.9.

RR-0009: *Allow static conversion to static discrete type of static discrete expression.* This request is considered under Section A.3.6. It might be worthwhile to go even further to allow general static scalar conversions, despite the problem of INTEGER(1.5). See Ada Comment 4709, which is reproduced here as Appendix C.1 on page 102.

RR-0010: *Allow the full declaration of a private type with discriminants to be a derived type.* This request is considered under Requirement R2.2-C(1). See also RR-0423.

RR-0011: *Expression 0**0 should not be 1 as this is an indeterminate form.* Rejected: too great a change from Ada 83. See Section 13.6. No change could be upward compatible.

RR-0012: *Mutation of types is needed for AI applications.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3. It isn't clear what capability is being requested.

RR-0013: *Allow task activation to occur at a higher priority than task execution.* This request is considered under Requirement R2.1-A(1). This request appears to be met by AI-00288, which requires that a task's activation occur at either its activator's priority or at its normal priority, whichever is higher.

RR-0014: *Need to call subprograms loaded in ROM.* This request is considered under Requirement R4.1-B(2). Ada 9X could ensure that pragma INTERFACE is usable to identify subprograms located in ROM.

RR-0015: *Allow task priorities to control all queuing/select decisions.* This request is considered under Requirement R5.2-A(1).

RR-0016: *Allow user-selectable task scheduling algorithms.* This request is considered under Requirement R5.2-A(1).

RR-0017: *Be able to treat an Ada object as an array of storage units.* This request is considered under Requirement R6.2-A(1). This RR gives an extensive example and discussion of difficulties in writing I/O packages for arbitrary data types.

RR-0018: *Need pre-elaborated constant arrays with variable-sized elements.* This request is considered under Requirement R6.4-A(1). This RR gives a careful and extensive discussion of the stated need.

RR-0019: *Allow types to specify finalization procedures for safely controlling use of collections.* This request is considered under Study Topic S4.2-A(2).

RR-0020: *Relative importance of functions may change during program execution, so priorities should be changeable.* This request is considered under Requirement R5.2-A(1).

RR-0021: *Need priority inheritance for server tasks.* This request is considered under Requirement R5.2-A(1).

RR-0022: *Need direct visibility of operators declared in another package.* This request is considered under Section A.2.3.

RR-0023: *Require TERMINATE alternative to terminate library tasks.* This request is considered under Requirement R2.1-A(1). This is already addressed by AI-00399.

RR-0024: *Need a way to decompose floating point numbers into mantissa/exponent.* This request is considered under Requirement R11.1-A(1).

RR-0025: *Allow overloading of the equality operator with different operand types.* This request is considered under Section A.3.9.

RR-0026: *Permit function parameters to have modes IN OUT and OUT.* Rejected: insufficient user benefit (OUT modes for functions). See Section 13.4.3.

RR-0027: *Improve generics so a generic report generator could be written.* This request is considered under Study Topic S4.4-A(1).

RR-0028: *Add a semicolon terminator to SEPARATE statement syntax.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4. Allowing an optional semicolon would not avoid confusion and requiring a semicolon would not be upward compatible or of much benefit.

RR-0029: *Allow use of OTHERS with named associations when the index constraint is determined by context.* This request is considered under Section A.2.5.

RR-0030: *Require subprogram specification to list non-local objects referred to.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0031: *Provide a way to test for a value in a non-contiguous set.* Rejected: insufficient user benefit (discontiguous subtypes). See Section 13.5.3.

RR-0032: *Allow grouping of variable declarations and related subprograms.* This request is considered under Section A.2.2.

RR-0033
This RR concerns two topics, each of which is treated separately. They are now listed.

RR-0033A: *Need to find the name of a raised exception.* This request is considered under Requirement R4.5-A(1).

RR-0033B: *Need to pass exceptions to subprograms and generic units.* This request is considered under Study Topic S4.4-A(1).

RR-0034: *Ada should use ISO 8859/1-9 (8-bit) character set.* This request is considered under Requirement R3.1-A(1).

RR-0035: *Allow generic units to be overloaded.* Rejected: insufficient user benefit (overloaded generic names). See Section 13.5.4.

RR-0036: *Allow exceptions to be grouped under a single name by allowing exception subtypes.* Rejected: insufficient user benefit (grouping exceptions). See Section 4.5-A(1).

RR-0037: *Allow tasks (i.e., delays) to execute using simulated time rather than a real-time clock.* This request is considered under Requirement R5.2-A(1). An application-controlled scheduling interface might allow tasks to execute in simulated time, although such a capability goes beyond the intent of the stated requirement.

RR-0038: *Allow expanded instead of simple names of subunits to be distinct.* This request is considered under Study Topic S4.3-C(1).

RR-0039: *Make it easier to access FORTRAN libraries.* This request is considered under Study Topic S11.2-A(1). This RR discusses quite extensively the problems of interfacing Ada with FORTRAN numerics subroutines.

RR-0040: *Need a way to determine the internal coding of enumeration values.* This request makes a useful suggestion for improvement in the ability to determine the representation of enumeration values. See Section 2.2.14 of this document.

RR-0041: *Allow overloaded subunits with respect to a common ancestor library unit.* This request is considered under Study Topic S4.3-C(1).

RR-0042: *Clarify the meaning of incorrect-order dependence and its effects.* This request is considered under Requirement R2.3-A(2).

RR-0043: *Make it easier and more portable to use assembler with Ada.* Rejected: not desirable to ease machine code insertion. See Section 13.1.1.

RR-0044: *There is no need to add unsigned integers to Ada.* Rejected: a contradictory requirement was made. See Section 13.2. See User Need U6.1-A.

RR-0045: *Allow/require extended precision for intermediate integer results.* This request is considered under Requirement R2.4-A(1). Extended precision is already allowed, but not required, by Ada; see 11.6(6) of the Standard.

RR-0046: *Allow testing in discontiguous ranges and create true sets.* Rejected: insufficient user benefit (discontiguous subtypes). See Section 13.5.3.

RR-0047: *Add TEXT_IO.GET functions.* This request is considered under Requirement R4.6-A(1). There might be some benefit in this suggestion to add value-returning subprograms to TEXT_IO given that a variable length string package is uniformly available.

RR-0048: *Extend static expressions to include representation attributes of composite types.* This request makes a useful suggestion for improvement in the ability to use representation attributes. See Section 2.2.4 of this document. Although the request is phrased in terms of allowing generic formal types to be used in static expressions, the example mainly shows a need to ensure that certain representation attributes can be used in static expressions.

RR-0049: *Allow special notation when the same name is on both sides of :=.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0050: *Provide multi-national and multi-byte characters.* This request is considered under Requirement R3.1-A(2).

RR-0051
This RR concerns three topics, each of which is treated separately. They are now listed.

RR-0051A: *Provide common mathematics packages.* This request is considered under Requirement R11.1-A(1).

RR-0051B: *Provide standard string manipulation packages.* This request is considered under Study Topic S10.4-A(1).

A: Numerical Listing of RRs

RR-0051C: *Provide packages for string edit functions.* This request is considered under Study Topic S10.4-A(2).

RR-0052: *Multiple derived types from same package do not give desired operations.* This request is considered under Study Topic S4.3-B(1). RR-0482 contains a better example and motivation.

RR-0053: *Allow aggregates for null records and arrays.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4. The RR points out that a null array aggregate can't be written if the component type is a private type for which no values are directly available. Nonetheless, it doesn't seem worthwhile to invent new notation just for these special cases.

RR-0054: *Do not add variable length strings to the language.* Rejected: a contradictory requirement was made. See Section 13.2. See Study Topic S10.4-A(1).

RR-0055: *Allow a subprogram body to be defined by renaming or generic instantiation.* This request is considered under Section A.4.1.

RR-0056: *Do not remove task entry families.* This request was met

RR-0057: *Need direct visibility to infix operators in another package.* This request is considered under Section A.2.3.

RR-0058: *Allow discontiguous subtypes of enumeration types.* Rejected: insufficient user benefit (discontiguous subtypes). See Section 13.5.3.

RR-0059: *Need an attribute for returning a representation's underlying value.* This request makes a useful suggestion for improvement in the ability to determine the representation of enumeration values. See Section 2.2.14 of this document.

RR-0060: *Allow inlining of subprograms from some but not all call sites.* This request makes a useful suggestion for improvement in controlling the effect of pragma INLINE. See Section 2.2.9 of this document.

RR-0061: *Make Long_Float and Short_Float required types.* This request is considered under Requirement R2.4-A(1).

RR-0062: *Ensure memory mapped devices are treated correctly by compilers.* This request is considered under Requirement R7.1-A(1).

RR-0063: *Protect tasks from being aborted while performing critical functions.* This request is considered under Requirement R5.3-A(1). Although this request is phrased in terms of protecting a task from being aborted, the underlying need is met by the Ada 9X requirement.

RR-0064: *Allow some form of subprogram callback.* This request is considered under Requirement R4.1-B(1).

RR-0065: *To improve reuse possibilities, allow rep clauses and various pragmas to be separated from the compilation unit to which they apply.* This request is considered under Study Topic S4.3-B(1).

RR-0066: *Reduce risks associated with erroneous execution/incorrect order dependences.* This request is considered under Requirement R2.3-A(2).

RR-0067: *Clarify/define technical terms used.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document. This RR provides some detailed comments that may be useful.

RR-0068: *The Standard should explicitly acknowledge that I/O support is optional for embedded systems.* This request is considered under Requirement R2.4-A(1). Implementation-dependent support for I/O functionality, particularly for implementations targeted to embedded systems, needs more attention.

RR-0069: *Allow subprograms and types to be added to a package without modifying the original package.* This request is considered under Study Topic S4.3-B(1).

RR-0070: *Allow user-defined assignment for limited types.* This request is considered under Study Topic S4.2-A(2).

RR-0071: *Improve support for heterogeneous distributed processing.* Rejected: a contradictory requirement was made. See Section 13.2. Dealing with heterogeneous distributed systems is beyond the scope of the revision (see Requirement R8.1-A(1)), but this RR discusses some of the issues that would have to be addressed otherwise.

RR-0072: *Prioritized queues and priority inheritance are needed for real-time applications* . This request is considered under Requirement R5.2-A(1).

RR-0073: *Allow visibility of names to be restricted within a program library.* This request is considered under Study Topic S4.3-C(1).

RR-0074: *Define a standard run-time support environment interface.* This request is considered under Requirement R5.2-A(1).

RR-0075: *Queue entries by task priority or FIFO based on application.* This request is considered under Requirement R5.2-A(1).

RR-0076: *Allow selection of entry calls from entry queues and open alternatives based on priorities.* This request is considered under Requirement R5.2-A(1).

RR-0077: *Provide stream I/O for digital signal processing.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4. The need for stream I/O to support digital signal processing is too narrow for motivating a

change, given the intended scope of the Ada 9X revision.

RR-0078: *Ada tasking is too complex, inflexible and inefficient.* These matters are addressed in Chapter 5 of the Requirements Document.

RR-0079: *TERMINATE alternative adds little value and is rarely used.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0080: *Derived types are clumsy.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3. The RR states that there are problems but does not identify them.

RR-0081: *Provide subprogram and package types.* This request is considered under Study Topic S4.1-A(1). Subprogram types is an obvious solution to this requirement. The RR is very short and provides no arguments explaining the need for package types.

RR-0082: *Allow declaration of objects of private types in visible package specification.* This request makes a useful suggestion for improvement in the ability to use private types before their full declaration. See Section 2.2.5 of this document.

RR-0083: *Provide asynchronous transfer of control via entry call/selective wait construct.* This request is considered under Requirement R5.3-A(1). The proposed solution is attractive.

RR-0084: *Specify standard conventions for using tasks that permit high-performance implementations.* This request is considered under Requirement R5.2-A(2). The intention here is to specify, in a real-time annex, the restrictions on task usage that allow tasks used for mutual exclusion to be implemented with special efficiency.

RR-0085: *Need to get the name of the current exception.* This request is considered under Requirement R4.5-A(1). This RR gives some valid reasons for accessing the name of an exception.

RR-0086: *Need to initialize a record component to the address of the record itself.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0087: *Allow software priorities to match/exceed hardware priorities.* This request is considered under Requirement R6.3-A(1).

RR-0088: *Problems associated with user-defined assignment.* This request is considered under Study Topic S4.2-A(2). This RR points out some problems to be addressed if user-defined assignment is added to the language.

RR-0089: *Provide facilities for I/O screen operations.* Rejected: low-level control of screen positioning via special commands to a terminal is outside the intended scope of the I/O packages.. See Section 4.6-A(1). Although there is a requirement for improved interactive text I/O functions, this request goes too far by asking for a screen management package. Text I/O is only intended to provide common, basic functionality.

RR-0090: *Allow some task entries to be visible, some not.* This request makes a useful suggestion for improvement in restricting the visibility of task entries. See Section 2.2.11 of this document.

RR-0091: *Don't specify the compilation process in the Standard.* This request is considered under Study Topic S4.3-C(1).

RR-0092: *Allow user-specified finalization* . This request is considered under Study Topic S4.2-A(2).

RR-0093: *Allow full declaration of deferred constants to be given in a package body.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4. Although the ability to defer the initialization of a constant to the body of a package would reduce the need for recompilation and although the RR proposes reasonable syntax, it is not clear that there is much demand for this change, and it would make new kinds of user errors possible (namely, accessing an uninitialized constant value).

RR-0094: *Make the multiple declaration rules more complete and consistent.* This request is considered under Requirement R2.2-B(1).

RR-0095: *Allow applicable units to be named in USE clauses and pragma ELABORATE.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0096
This RR concerns three topics, each of which is treated separately. They are now listed.

RR-0096A: *Permit renaming an enumeration literal as a character literal.* This request is considered under Section A.2.3.

RR-0096B: *Allow a procedure body to be provided by a renaming declaration.* This request is considered under Section A.4.1.

RR-0096C: *Allow the full declaration of a private type to be provided by a renaming declaration.* This request is considered under Section A.4.2.

RR-0097: *Allow/require explicit action to get default parameter value.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0098: *Generalize incomplete typing for types other than access or private.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4. The limitation mentioned here may be satisfied by changes made under Study Topic S4.3-B(1).

RR-0099: *Explicit type conversions should be allowed in static expressions.* This request is considered under Section A.3.6. The RR gives an example of a problem caused by the current rules.

A: Numerical Listing of RRs

RR-0100: *Allow constants to use default values to get value.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0101
This RR concerns two topics, each of which is treated separately. They are now listed.

RR-0101A: *Allow exceptions to be grouped under a single name.* Rejected: insufficient user benefit (grouping exceptions). See Section 4.5-A(1).

RR-0101B: *Need to pass exceptions as parameters to generic units and subprograms.* This request is considered under Study Topic S4.4-A(1). Most of this RR deals with the ability to group exception names.

RR-0102: *Provide explicit remainder operator for real numbers.* This request is considered under Requirement R11.1-A(1).

RR-0103
This RR concerns two topics, each of which is treated separately. They are now listed.

RR-0103A: *Allow unchecked conversion for IN OUT and OUT parameters.* This request makes a useful suggestion for improvement in the treatment of subprogram parameters. See Section 2.2.3 of this document.

RR-0103B: *Provide efficient means of reading large data structures in chunks.* This request is considered under Requirement R6.2-A(1). This problem could also be solved by providing an appropriate FORM parameter when opening a file, so a large data structure would be read or written in several blocks, thereby using smaller internal buffers.

RR-0104: *Prohibit access to a task outside its master.* This request is considered under Section A.1.1.

RR-0105: *Allow application to set/adjust clocks.* This request is considered under Requirement R5.1-A(1). The ability to adjust the elapsed time or time-of-day clocks is implicit in the stated requirement.

RR-0106: *Provide asynchronous transfer of control.* This request is considered under Requirement R5.3-A(1). This RR contains a good rationale and good examples dealing with the need for asynchronous transfer of control.

RR-0107: *Allow application to specify clock timing interval if hardware allows this flexibility.* This request is considered under Requirement R5.1-A(1).

RR-0108: *Need to be able to wake up a task at a particular local time.* This request is considered under Requirement R5.1-B(1). Provides a good discussion of some of the issues.

RR-0109: *Provide Ada semantics that are helpful when dealing with a single distributed Ada program.* This request is considered under Requirement R8.1-A(1). The request for distribution across heterogeneous processors is not met. This RR, however, gives a good discussion of some of the key problems that make use of Ada 83 more difficult than necessary for distributed processing.

RR-0110: *Provide explicit control over placement of and access to data in different types or regions of memory.* This request is considered under Requirement R6.4-A(1).

RR-0111: *Provide explicit support for fault tolerance and recovery.* This request is considered under Requirement R8.1-A(1). This is covered by item 2 of the Requirement.

RR-0112: *Provide user support for controlled space reclamation.* This request is considered under Requirement R4.2-A(1). This RR provides an example user interface for controlling storage allocation and reclamation.

RR-0113: *Ensure that there are no storage ''leaks''.* This request is considered under Requirement R4.2-A(1). This RR gives some examples of how storage leaks can occur.

RR-0114: *Allow an address clause for each task instance, and not just on the type.* This request is considered under Requirement R6.3-A(2). Meeting this requirement should solve the problem underlying this RR.

RR-0115: *Provide better interrupt handling model.* This request is considered under Requirement R6.3-A(1). This RR contains a good discussion of current problems in dealing with interrupts.

RR-0116: *User-modifiable priorities needed for mode change and graceful degradation.* This request is considered under Requirement R5.2-A(1). This RR gives brief examples supporting the stated need.

RR-0117: *Provide pre-elaboratable constructs.* This request is considered under Requirement R8.2-A(1). This RR presents a set of possible rules defining units that can be pre-elaborated.

RR-0118: *Provide a user-specified storage reserve for STORAGE_ERROR recovery.* This request is considered under Requirement R4.2-A(1). This capability could be automatically made available if Ada 9X allowed user-defined storage management operations to be written.

RR-0119: *Need synchronized reference to elements of shared composite objects.* This request is considered under Requirement R7.1-A(1). This RR provides a good discussion of some problems concerning the use of memory locations shared among tasks, e.g., memory-mapped I/O and guarding against optimizations that remove references to volatile memory locations.

RR-0120: *Allow users to defer the signalling of STORAGE_ERROR when space is exhausted.* This request is considered under Requirement R4.2-A(1). The problem can be solved in its full generality only by customizing a storage allocator.

RR-0121: *Provide more user control over scheduling decisions.* This request is considered under Requirement R5.2-A(1).

RR-0122: *Permit an implementation to reject some integer types as array indexes.* This request is considered under Requirement R2.2-A(1).

RR-0123: *Provide initialization values to tasks at startup.* This request is considered under Study Topic S7.2-A(1). This RR provides an extensive discussion and examples illustrating the problem and a possible solution.

RR-0124: *Ensure that code dependent on task scheduling algorithms is portable.* This request is considered under Requirement R5.2-A(1). Although this RR discusses AI-00594, which is not an approved AI, the concern of the RR is reflected in its title. This concern has been addressed by the notion of a real-time Annex for Ada 9X, since one purpose of the annex is to improve the performance portability of code. In addition, the requirement for user-controlled scheduling algorithms makes portability more possible.

RR-0125: *Introduce object-oriented inheritance into the language.* This request is considered under Study Topic S4.3-B(1).

RR-0126: *Allow underscore before ''E'' in exponents.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0127: *Allow real number output in non-decimal bases.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0128: *Provide subprograms as parameters to subprograms and entries.* This request is considered under Requirement R4.1-B(1).

RR-0129: *Allow default initialization to be specified for any non-limited type.* This request makes a useful suggestion for improvement in default initialization capabilities. See Section 2.2.2 of this document.

RR-0130: *Replace DEFAULT_xy variables in Chapter 14 by functions.* This request is considered under Requirement R4.6-B(1). RR-0484 proposes a better change.

RR-0131: *In a qualified expression, should have visibility of the enumeration literals of the qualifying type.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0132: *Allow optional WHEN <condition> on RAISE statement for consistency with EXIT statement.* This request is considered under Section A.3.4.

RR-0133: *Allow a task component of an array to get its index.* This request is considered under Study Topic S7.2-A(1). This RR explicitly cites an example for a massively parallel architecture.

RR-0134: *Require re-evaluation of entry'count on abandoned entries.* Rejected: too great a change from Ada 83. See Section 13.6. It is not clear that this would be easy or efficient to implement reliably due to race conditions. Once evaluation of a select statement has begun, no entry calls could be abandoned until one alternative has been selected.

RR-0135: *Catenation should not raise CONSTRAINT_ERROR for intermediate results.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0136: *Provide support for bit-field operations such as shift, rotate.* This request is considered under Requirement R6.1-A(1).

RR-0137: *Standardize bit storage/order conventions.* This request is considered under Requirement R2.4-A(1).

RR-0138: *Need full-sized unsigned integers.* This request is considered under Requirement R6.1-A(1).

RR-0139: *Provide shift and rotate operations for boolean arrays.* This request is considered under Requirement R6.1-A(1). The requirement supplies the requested functionality.

RR-0140: *Provide support for object-oriented programming.* This request is considered under Study Topic S4.3-B(1).

RR-0141: *Allow WHEN <condition> on RAISE statements.* This request is considered under Section A.3.4.

RR-0142: *Reduce cases where recompilation of subunits is needed.* This request is considered under Study Topic S4.3-A(1). This RR gives some examples of the kinds of program changes that should not force recompilation.

RR-0143: *Document implementation dependences.* This request is considered under Study Topic S9.1-A(1).

RR-0144: *Require support for fixed point arithmetic even if floating point hardware is not present.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3. It is not clear what language change, if any, is being requested.

RR-0145: *Provide a way to get exception name from WHEN OTHERS handlers.* This request is considered under Requirement R4.5-A(1). This RR references a POSIX requirement for the ability to print the name of an exception from within an OTHERS handler.

RR-0146: *Support for file/record locking.* Rejected: This is too specialized a capability to require for every implementation. See Section 13.1.

RR-0147: *Add support for ISAM.* Rejected: Although an ISAM package might be useful, there are too many other higher priority requirements that should be addressed. See Section 13.1.

A: Numerical Listing of RRs

RR-0148: *Provide support for extended and graphic characters (256 ASCII set).* This request is considered under Requirement R3.1-A(1).

RR-0149: *Provide a keyboard input/output package.* This request is considered under Requirement R4.6-A(1).

RR-0150: *Provide ''chaining'' of different programs to reduce memory requirements.* Rejected: not a language issue. See Section 13.7.

RR-0151: *Need standard support for priority interrupts.* This request is considered under Requirement R6.3-A(1).

RR-0152: *Allow e.g., a < b < c which would mean a < b AND b < c.* Rejected: too great a change from Ada 83. See Section 13.6. The RR proposes the new idea of n-ary operators.

RR-0153: *Private part foils separation of specification and implementation.* Rejected: Much of the requested functionality can be obtained by completing an incomplete type in a package body. See Section 13.1.

RR-0154: *Subunits should not have to be at the outermost compilation unit level.* Rejected: The ability to declare a subunit in a nested block would require extra complications in requiring that all enclosing blocks be named. Allowing subunit declarations in nested units but not in blocks would seem to be a non-uniformity, so there is no easy way to provide the requested capability. See Section 13.1.

RR-0155: *Define RANGE attribute for scalar types.* This request is considered under Section A.3.3.

RR-0156: *A negative literal should be allowed wherever a literal is allowed.* This request is considered under Section A.3.12.

RR-0157: *Allow renaming when defining a subprogram body.* This request is considered under Section A.4.1. This RR gives examples showing the usefulness of the proposed capability.

RR-0158: *Allow multi-way conditional and timed entry calls.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3.

RR-0159: *Add standard package of general file system functions.* This request is considered under Requirement R4.6-B(1).

RR-0160: *Allow user-defined assignment for limited types.* This request is considered under Study Topic S4.2-A(2).

RR-0161: *Allow default initialization for any non-limited type.* This request makes a useful suggestion for improvement in default initialization capabilities. See Section 2.2.2 of this document.

RR-0162: *Provide a clean interface to a SORT package.* Rejected: Providing attributes for use with a standard interface to a sort package would be useful in information system applications, but other changes were judged to have higher priority. See Section 13.1.

RR-0163: *Need support for variable-length strings with appropriate equality and assignment operations.* This request is considered under Study Topic S10.4-A(1).

RR-0164: *Provide multitasking terminal I/O in TEXT_IO.* This request is considered under Requirement R4.6-A(1).

RR-0165: *Allow parameter constraint violations to be compile-time errors.* This request is considered under Study Topic S2.3-A(1).

RR-0166: *Allow definition of the literal representations of an abstract data type.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3.

RR-0167: *Allow classes of abstract data types.* This request is considered under Study Topic S4.3-B(1).

RR-0168: *Allow implicitly-invoked finalization code for storage management.* This request is considered under Study Topic S4.2-A(2).

RR-0169: *Allow ''null'' procedures for actual or default generic formal subprogram values.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4. This problem is discussed in more detail in the Ada 9X report, ''Ada Support for Software Reuse'' [6].

RR-0170: *Permit or provide alternate scheduling algorithms.* This request is considered under Requirement R5.2-A(1).

RR-0171: *Allow target-dependent code (including rep clauses) to be separate from other code.* This request is considered under Study Topic S4.3-B(1).

RR-0172: *Make import and export of types easier.* This request is considered under Study Topic S4.3-B(1).

RR-0173: *Allow a rendezvous with a higher-level entity, i.e., a set of tasks.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3.

RR-0174: *Allow packages to be generic with respect to concurrency protection.* This request is considered under Study Topic S4.3-B(1). This is a good example of a form of specialization that is often needed.

RR-0175: *Define interface between compiler- and target-specific run-time system aspects.* This request is considered under Requirement R5.2-A(1). The possibility of a standardized RTS interface is discussed under this requirement.

RR-0176: *Document run-time system performance and memory allocation strategies.* This request is considered under Study Topic S9.1-A(1).

RR-0177: *Standardize interface between compiler and library for configuration management.* This request is considered under Study Topic S4.3-C(1).

RR-0178: *Problems with name clashes with big program libraries.* This request is considered under Study Topic S4.3-C(1).

RR-0179: *The treatment of interrupts is too implementation-dependent.* This request is considered under Requirement R6.3-A(1). Several problems are discussed in detail in this RR.

RR-0180: *There is a need for procedures as parameters for X-Windows, etc.* This request is considered under Requirement R4.1-B(1).

RR-0181: *Need standard means of communicating between Ada programs.* This request is considered under Requirement R8.1-A(1).

RR-0182: *Define visibility limits for parts of a program running on different processors.* This request is considered under Requirement R8.1-A(1).

RR-0183: *Asynchronous inter-task communication is not available.* This request is considered under Requirement R5.4-A(1).

RR-0184: *Need user-defined assignment operator for limited private type.* This request is considered under Study Topic S4.2-A(2).

RR-0185: *General Ada rendezvous is slow; semaphores would be better.* This request is considered under Requirement R5.2-A(2).

RR-0186: *It is difficult to write an entire operating system in Ada.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3. The request asks for additional ways to refer to a task and to control it, but gives no examples of what deficiencies are to be remedied or what additional control is thought to be needed.

RR-0187: *Need to allow unsigned enumeration representation specifications.* This request is considered under Requirement R2.4-A(1). This RR suggests that the representation of enumeration values cannot be controlled adequately since the treatment of sign extension for negative literals is not adequately controlled by the standard. It is not clear that this complaint is justified, but it should be given consideration.

RR-0188: *Embedded applications need unsigned integers and bit-wise logical operations on integer types.* This request is considered under Requirement R6.1-A(1).

RR-0189: *Standard should include a floating-point math library interface.* This request is considered under Requirement R11.1-A(1).

RR-0190: *Allow use of a base type within a generic unit.* This request is considered under Study Topic S4.4-A(1).

RR-0191: *Fixed point model numbers should include the bounds of the type definition.* This request is considered under Requirement R2.2-B(1).

RR-0192: *Need ability to change priorities during mode change and for graceful degradation.* This request is considered under Requirement R5.2-A(1).

RR-0193: *Allow priority queues, priority inheritance, and prioritized treatment of open select alternatives.* This request is considered under Requirement R5.2-A(1).

RR-0194: *Disallow referencing a task from outside its master.* This request is considered under Section A.1.1.

RR-0195: *Need interrupt address per task, not task type.* This request is considered under Requirement R6.3-A(2). Meeting this requirement should solve the problem underlying this RR.

RR-0196: *Endorsement of RR-0083.* This request is considered under Requirement R5.3-A(1). This RR endorses the solution suggested in RR-0083 and repeats some material found in [8].

RR-0197: *For access types, parameter mode IN should mean the designated object cannot be modified.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0198: *Allow positional aggregate for single-component aggregate.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0199: *Allow IF, CASE, and SELECT constructs to be named.* This request is considered under Section A.3.13.

RR-0200: *Allow optional when_clause on RAISE and RETURN statements.* This request is considered under Section A.3.4.

RR-0201
This RR concerns two topics, each of which is treated separately. They are now listed.

RR-0201A: *Liberalize overloading of operators to other character sequences.* Rejected: too much implementor change for the payoff (user-defined operator syntax). See Section 13.5.1.

RR-0201B: *Overload the assignment operation.* This request is considered under Study Topic S4.2-A(2).

RR-0202: *Relax parameter mode rules for limited types that have an assignment operation.* This request is considered under Study Topic S4.2-A(2). These problems will be addressed by allowing user-defined assignment for limited types.

RR-0203: *Allow finalization code for packages and tasks.* This request is considered under Study Topic S4.2-A(2).

RR-0204: *Clarify which fixed point operators are predefined.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document. This RR proposes an improvement to the Standard's Appendix C.

RR-0205: *Allow program unit name on PRIVATE, BEGIN, and EXCEPTION.* This request is considered under Section A.3.13.

A: Numerical Listing of RRs

RR-0206: *Paragraph numbers should be included in the cross references.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document.

RR-0207: *Add TEXT_IO support with Exists function and Append procedure.* This request is considered under Requirement R4.6-B(1).

RR-0208: *Need ability to initiate TEXT_IO, DIRECT_IO, and SEQ_IO operations without waiting for completion.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4. The need for high level asynchronous I/O is not sufficiently great to warrant a change to implementations.

RR-0209: *Require the compiler to report certain-to-be-raised exceptions.* This request is considered under Study Topic S2.3-A(1).

RR-0210: *Need more pragmas for software maintenance to MIL standards.* Rejected: not a language issue. See Section 13.7.

RR-0211: *Require compilers to report unrecognized or incorrect pragmas.* This request is considered under Study Topic S2.3-A(1).

RR-0212: *Allow assignment to record discriminant like other components.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0213: *Need to be able to find out if an implementation rounds up or down.* This request is considered under Requirement R2.4-A(1).

RR-0214: *Require that a subprogram parameter be used within the body.* Rejected: Such a change would be an inconvenience during program development. See Section 13.1.

RR-0215: *Clarify termination of tasks dependent on library packages.* This request is considered under Requirement R2.1-A(1). This problem is addressed by AI-00399.

RR-0216: *Require that each task entry have at least one accept statement.* This request is considered under Requirement R9.3-A(1). Requiring at least one accept statement for each entry may be a reasonable project coding convention that should be enforceable by compilers.

RR-0217: *Require that a parameter of an entry be used within an accept.* Rejected: Such a change would be an inconvenience during program development. See Section 13.1.

RR-0218: *Make the implementation find a good library-unit elaboration order.* This request is considered under Section A.2.1. The problem is relevant to a revision of pragma ELABORATE.

RR-0219: *Provide a way to get the name of the last raised exception, including an out-of-scope exception.* This request is considered under Requirement R4.5-A(1).

RR-0220: *Need way to get the internal code associated with enumeration values.* This request makes a useful suggestion for improvement in the ability to determine the representation of enumeration values. See Section 2.2.14 of this document.

RR-0221: *Need to write common code for group of exception handlers.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0222: *Need additional predefined packages for process control/communication.* This request is considered under Requirement R8.1-A(1).

RR-0223: *Need to add inheritance to support object-oriented programming.* This request is considered under Study Topic S4.3-B(1).

RR-0224: *Add communication support required for distributed systems.* This request is considered under Requirement R8.1-A(1).

RR-0225: *Ensure floating point representation with desired accuracy is used.* This request is considered under Study Topic S11.1-B(1).

RR-0226: *Need standardized support for improved library management capabilities.* This request is considered under Study Topic S4.3-C(1). Presents a reasonable discussion of library management needs.

RR-0227: *Allow generic parameterization with static numeric quantities.* This request is considered under Study Topic S4.4-A(1).

RR-0228: *Allow generic parameterization with exceptions.* This request is considered under Study Topic S4.4-A(1).

RR-0229: *Need to hide the range of a scalar type and the initial value of an object to ensure these values are not used directly by programmers.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0230: *Allow initialization to be associated with any type definition.* This request makes a useful suggestion for improvement in default initialization capabilities. See Section 2.2.2 of this document.

RR-0231: *Allow a rename definition of a subprogram body.* This request is considered under Section A.4.1. Examples of inadequate workarounds are given.

RR-0232: *Need to allow direct visibility of operators in packages.* This request is considered under Section A.2.3.

RR-0233: *Pragma ELABORATE should be transitive.* This request is considered under Section A.2.1.

RR-0234: *''Sub-null'' ranges are of little value and an implementation burden.* This request is considered under Section A.1.2.

RR-0235: *Need support for interactive terminal input/output.* This request is considered under Requirement R4.6-A(1).

RR-0236: *Reduce implementation-dependent behavior, or at least, ensure it is documented whenever possible.* This request is considered under Requirement R2.4-A(1).

RR-0237: *Make separate compilation independent of a particular library model.* This request is considered under Study Topic S4.3-C(1).

RR-0238: *Allow access values to designate read-only memory.* This request is considered under Requirement R6.4-A(1).

RR-0239
This RR concerns two topics, each of which is treated separately. They are now listed.

RR-0239A: *Renaming an enumeration type should make literals visible.* This request is considered under Section A.2.3.

RR-0239B: *A renamed type cannot be used in an actual parameter type conversion.* This request makes a useful suggestion for improvement in the treatment of subprogram parameters. See Section 2.2.3 of this document.

RR-0240: *Non-sliding aggregates and slices in component associations.* This request is considered under Section A.3.11. The RR points out inconsistencies between assignment and component association.

RR-0241: *Need easier and more efficient support for mutual exclusion.* This request is considered under Requirement R5.2-A(2).

RR-0242: *Require compilation warnings for potential run-time errors.* This request is considered under Study Topic S2.3-A(1).

RR-0243: *Allow/require elaboration prior to run time.* This request is considered under Requirement R8.2-A(1).

RR-0244
This RR concerns two topics, each of which is treated separately. They are now listed.

RR-0244A: *Require pre-elaboration of some constructs.* This request is considered under Requirement R8.2-A(1).

RR-0244B: *Flag run-time errors at compile-time when possible.* This request is considered under Study Topic S2.3-A(1).

RR-0245: *Change Standard to encourage pre-elaboration.* This request is considered under Requirement R8.2-A(1).

RR-0246: *Ensure that constant declarations are not elaborated at run time when initialized with static expressions.* This request is considered under Requirement R8.2-A(1). The problem addressed here is pre-elaboration, although the proposed solution is too drastic.

RR-0247: *Don't initialize access variables by default to NULL.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0248: *Allow users to specify locations for discriminants that are outside record values.* Rejected: The RR does not provide sufficient justification for allowing non-local record discriminants. See Section 13.1.

RR-0249: *'First and 'last for null ranges are defined oddly.* This request is considered under Section A.1.2. This RR gives a specific example of a problem.

RR-0250: *Define clearer notation for expressing null ranges.* This request is considered under Section A.1.2.

RR-0251: *Invent new notations to distinguish function call, array reference, and conversions.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0252
This RR concerns five topics, each of which is treated separately. They are now listed.

RR-0252A: *Ensure support for IEEE floating point standard; allow full use of machine characteristics.* This request is considered under Study Topic S11.1-B(1).

RR-0252B: *Programmer needs to know/control whether rounding or truncation is used in real calculations.* This request is considered under Study Topic S11.1-B(1).

RR-0252C: *Ensure programmer can choose appropriate floating point representation.* This request is considered under Study Topic S11.1-B(1).

RR-0252D: *Fixed point type should include the bounds of the range definition.* This request is considered under Requirement R2.2-B(1).

RR-0252E: *Provide a floating point model that reflects actual machine architecture.* This request is considered under Study Topic S11.1-B(1).

RR-0253: *DIGITS and DELTA approach leads to inefficiency, non-portability.* Rejected: This RR does not reflect a correct understanding of the efficiency impacts of DIGITS and DELTA specifications. See Section 13.1.

RR-0254: *Too much freedom is allowed with respect to exceptions and intermediate expression results.* This request is considered under Requirement R9.1-A(2).

RR-0255: *Provide a function for returning the value of the next floating point number.* This request is considered under Requirement R11.1-A(1).

RR-0256: *Fixed-point approach with range and delta is not what is needed.* Rejected: Fixed point representations can be completely controlled in Ada 83 with proper use of 'SMALL and 'SIZE representation clauses. See Section 13.1.

RR-0257: *Ensure that BOOLEAN and BYTE arrays can be tightly packed.* This request is considered under Requirement R2.1-A(1). AI-00555, which has been approved by the Ada Rapporteur Group, specifies that arrays of boolean components must be packed with no gaps. AI-00556 addresses the problem of arrays of bytes, but has not yet been approved.

RR-0258: *Need access values that point to declared objects.* This request is considered under Re-

quirement R6.4-A(1). The purpose behind this request is to be able to establish static data structures linked by pointers.

RR-0259: *Incomplete type declarations are dangerous and unnecessary.* Rejected: not a language issue. See Section 13.7. This request reflects an incorrect understanding of Ada.

RR-0260: *The Standard is unclear in various ways.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document. The RR contains several useful suggestions. However, the submitter wants the Standard to be more tutorial, which is probably not possible in a document intended to serve as the specification for a language.

RR-0261: *Need compile-time warnings for access before elaboration errors.* This request is considered under Study Topic S2.3-A(1).

RR-0262: *Do not require existence of subunit for body stubs.* Rejected: not a language issue. See Section 13.7. A CASE tool can provide the requested functionality.

RR-0263: *CONSTRAINT_ERROR is too broadly defined.* Rejected: This issue was given thorough consideration in the original design. Insufficient evidence is given in this RR to justify reconsidering the decision. See Section 13.1.

RR-0264: *Discriminants need to stand out more.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3.

RR-0265: *Allow implementations to short-circuit in general, forget AND THEN.* Rejected: not a language issue. See Section 13.7. This request reflects an incorrect understanding of Ada.

RR-0266: *Operator overloading is dangerous.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0267: *The Standard is confusing in distinguishing specifications and declarations.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document. The submitter wants the Standard to be more tutorial.

RR-0268: *Separation of specification and body is not worth it.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0269: *Make subprograms not recursive by default.* Rejected: too great a change from Ada 83. See Section 13.6. This is a substantive change to the language, not just a change to a note as implied by the RR.

RR-0270: *Allow specification of read-only data from a package.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0271: *Distinguish storage classes for variables with key words like CONTROLLED or STATIC.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0272: *Limited types are of little true value.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0273: *There are problems with private types in the language.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0274: *The visibility rules could be explained more clearly.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document. The submitter wants the Standard to be more tutorial.

RR-0275: *Error-prone and counter-intuitive aspects of RENAMES.* This request is considered under Requirement R2.2-B(1).

RR-0276: *Need user specified accuracy and precision control over timing.* This request is considered under Requirement R5.1-A(1).

RR-0277: *Inappropriate wording.* Rejected: The wording (in 8.6(1), not 8.8(1) as in the RR) is acceptable. For 9(5) the comment refers to a note, which is worded acceptably. See Section 13.1.

RR-0278: *Tasking model should support common scheduling disciplines more easily.* This request is considered under Requirement R5.2-A(2).

RR-0279: *If tasks are not used, the run-time system and compiled code should not include code for tasking support.* This request is considered under Requirement R2.2-A(1).

RR-0280: *Short delays are too inefficient; Calendar time unnecessary; timing performance must be documented.* This request is considered under Requirement R5.1-A(1). The general tenor of this request is that Ada's timing model is not appropriate for embedded real-time systems, but the purpose of the requirements is to ensure that the Ada 9X model indeed is appropriate.

RR-0281: *Confusing treatment of term ''delay statement''.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document.

RR-0282: *Ada program structure hides important context information.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3.

RR-0283: *Need convenient way to set global compilation parameters.* This request is considered under Study Topic S4.3-C(1).

RR-0284: *Machine-code insertions are unreadable; replace with INLINE macros.* Rejected: not desirable to ease machine code insertion. See Section 13.1.1.

RR-0285: *Minimize the need for run-time elaboration.* This request is considered under Requirement R8.2-A(1).

RR-0286
This RR concerns four topics, each of which is treated separately. They are now listed.

RR-0286A: *Embedded system users need the ability to control timer utilities.* This request is considered under Requirement R5.2-A(1).

RR-0286B: *Embedded system user may need access to interrupts that are also used by the run-time system.* This request is considered under Requirement R5.2-A(1).

RR-0286C: *Run-time system should avoid entering privileged mode.* This request is considered under Requirement R5.2-A(1).

RR-0286D: *Interrupts should be handled with a procedure model, not a task model.* This request is considered under Requirement R6.3-A(1).

RR-0287: *Make access types point directly to designated object.* This request is considered under Requirement R2.4-A(1). In some implementations, access values point to dope vectors rather than the designated object. This causes unnecessary implementation-dependence when interfacing with other languages.

RR-0288: *Integrate representation clause information with declarations.* Rejected: too much implementor change for the payoff. See Section 13.5.

RR-0289: *Need multiple views of a record structure even when no discriminant is present.* This request is considered under Requirement R6.2-A(1). Unchecked conversion is not the answer to this problem, since UC can't be used as the target in an assignment and copying is too inefficient.

RR-0290: *The syntax used in record representation clauses is hard to read.* Rejected: too much implementor change for the payoff. See Section 13.5.

RR-0291: *Clarify whether use of an address clause causes storage to be initialized.* This request is considered under Requirement R6.4-A(1).

RR-0292: *Section 13.6 of the standard has no semantic content.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document. The RR notes correctly that the section is, in essence, just a note and perhaps should be so titled.

RR-0293: *Allow access values to point to declared objects.* This request is considered under Requirement R6.4-A(1). No examples are given.

RR-0294: *I/O packages are not suitable for embedded applications; make Chapter 14 optional.* This request was met — I/O is already not required if it can't be supported by the target platform.

RR-0295: *Create TEXT_IO.PUT_LINE for types other than string (make like PUT).* This request is considered under Requirement R4.6-B(1). The operations called for here arguably improve the uniformity and teachability of TEXT_IO, but might also be considered to clutter the definition.

RR-0296: *Make predefined I/O packages optional if appropriate.* This request was met — I/O is not required if it is not appropriate for the platform.

RR-0297: *LOW_LEVEL_IO was a bad idea; remove this package from the language.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0298: *Clarify classes of objects usable as attribute prefixes.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document. The submitter wants the Standard to be more tutorial.

RR-0299: *Make everything in the Standard ''part of the standard''.* Rejected: Many readers find the extra material useful. See Section 13.1.

RR-0300: *Use an LR grammar to define the syntax of the language.* Rejected: a contradictory requirement was made. See Section 13.2. Requirement R2.1-B(1) discourages such changes.

RR-0301: *The wording concerning checking for consistency between compilations can be improved.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document. The RR suggests a helpful rewording.

RR-0302: *The language should define literals for values of type ADDRESS.* This request is considered under Requirement R2.4-A(1).

RR-0303: *Allow reading of OUT parameters.* This request is considered under Section A.3.10.

RR-0304: *Define RANGE attribute for scalar types.* This request is considered under Section A.3.3.

RR-0305: *Clarify wording of FOR loop completion.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document.

RR-0306: *Need to be able to start processing at a particular time of day.* This request is considered under Requirement R5.1-B(1).

RR-0307: *Allow completion of private declarations to be in the package body.* This request is considered under Study Topic S4.3-A(1). The RR gives a reference to a paper justifying a conclusion that efficient code can be generated even if a private type's full declaration is given in a package body.

RR-0308: *Add libraries for array processing.* This request is considered under Requirement R11.1-A(1).

RR-0309: *Ensure all cross references are complete and correct.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document.

A: Numerical Listing of RRs

RR-0310: *Need convenient way to pad with blanks in string assignments.* This request is considered under Study Topic S10.4-A(1). A varying-length string library might obviate the need for this functionality.

RR-0311: *Generalize character set for 8-bit characters.* This request is considered under Requirement R3.1-A(1).

RR-0312: *Generalize case statement to decision table.* Rejected: too much implementor change for the payoff. See Section 13.5.

RR-0313: *Allow deferred constants of arbitrary (i.e., non-private) types.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0314: *Define minimum-quality error diagnostics in the standard.* Rejected: not a language issue. See Section 13.7.

RR-0315: *Allow integer type names that indicate representation size, e.g., INTEGER_32, to improve portability.* This request is considered under Requirement R2.4-A(1). This RR also recommends that the standard state explicitly that the length of LONG_INTEGER not be less than the length of INTEGER, with similar constraints imposed on SHORT_INTEGER.

RR-0316: *Improve interrupt handling, e.g., with interrupt procedures.* This request is considered under Requirement R6.3-A(1).

RR-0317: *Augment Ada's looping: over reals, list items, etc.* This request makes a useful suggestion for improvement in iteration constructs. See Section 2.2.12 of this document.

RR-0318: *Make a machine-readable version of the Standard available (with embedded mark-up).* This request is considered under Requirement R2.1-C(1).

RR-0319: *Remove arbitrary language restrictions, improve orthogonality.* This request is considered under Requirement R2.2-C(1). The RR does not give any specific suggestions, but the general intent of the RR is consistent with the requirement for generality.

RR-0320: *Generalize case statement for other types, including REAL.* Rejected: too much implementor change for the payoff. See Section 13.5.

RR-0321: *Permit anonymous array and record declarations for record components.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0322: *Do not add any new reserved words to the language.* Rejected: This matter will be resolved by the Mapping/Revision Team, and there is no direction in the Requirements Document. Note, however, that the magnitude of the requirements is such that it is not likely to be practical to meet this request. See also the Upward Compatibility guideline on page 5 of the Requirements Document. See Section 13.1.

RR-0323: *Generalize slice for multidimensional arrays.* Rejected: insufficient user benefit (multi-dimensional slices). See Section 13.4.2.

RR-0324: *Add more flexible support for string manipulation.* This request is considered under Study Topic S10.4-A(2). The RR suggests incorporating string manipulation operations that are supported in ICON, PL/I, and REXX.

RR-0325
This RR concerns two topics, each of which is treated separately. They are now listed.

RR-0325A: *Allow implementations to enforce local coding standards.* This request is considered under Requirement R9.3-A(1).

RR-0325B: *Allow implementations to experiment with supersets.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0326: *Use a different syntax production style.* Rejected: a contradictory requirement was made. See Section 13.2. This RR suggests that the Ada syntax productions should provide more information about program legality and suggests that an attribute grammar should be used. This kind of stylistic change has been ruled out of scope by Requirement R2.1-B(1).

RR-0327: *Add varying length strings to the language.* This request is considered under Study Topic S10.4-A(1).

RR-0328: *Require compilers to report questionable uses of the language.* This request is considered under Requirement R9.3-A(1). This RR does not list any specific questionable uses.

RR-0329: *Using a deferred constant before it has a value.* Rejected: The apparent problem raised by this request does not exist. The example given in the RR is illegal by 7.4.1(3). See Section 13.1.

RR-0330: *Allow national characters in literals, comments, and identifiers.* This request is considered under User Need U3.1-A. This request is addressed by Requirements R3.1-(A1-5).

RR-0331: *Need predefined LONG_CHARACTER (16 bits) and LONG_LONG_CHARACTER (32).* This request is considered under Requirement R3.1-A(2).

RR-0332: *Provide unsigned integer capability.* This request is considered under Requirement R6.1-A(1). This RR provides a fairly extensive discussion of the need and the language design difficulties.

RR-0333: *More precise definition of TEXT_IO is needed, less implementation freedom.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3. The RR says that there are problems but does not identify them.

RR-0334: *Need to specify task parameters giving a task its work domain, e.g., to process part of an array.* This request is considered under Study Topic S7.2-A(1).

RR-0335: *Effect of abort statement is too implementation-dependent.* This request is considered under Requirement R5.3-A(1). The requirement addresses the problem raised in the RR.

RR-0336: *Allow array type definitions in records; nice for array-of-array case.* Rejected: insufficient user benefit (anonymous arrays as record components). See Section 13.4.4. LSN-222 discusses the potential complexity of allowing this capability. See Language Study Notes, 1983, available from the Ada Information Clearinghouse.

RR-0337: *Provide some form of user-modifiable priorities.* This request is considered under Requirement R5.2-A(1). Both mode changes and graceful degradation are mentioned in examples.

RR-0338: *Provide pointers to static objects and safe conversion between ADDRESS values and access values.* This request is considered under Requirement R6.4-A(1). Examples include large data structures such as maps residing in ROM. The use of unchecked conversion is too implementation-dependent and unsafe because addresses and access values do not necessarily have the same representation.

RR-0339: *Support sorting in extended alphabets.* Rejected: There does not appear to be any solution at the language level. See the discussion following Requirement R3.1-A(1). See Section 13.1.

RR-0340: *Allow optional simple name on CASE, IF, and SELECT statements.* This request is considered under Section A.3.13.

RR-0341: *Allow discriminant value in record aggregate to be non-static.* This request is considered under Requirement R2.2-C(1). The RR makes a useful suggestion for removing a restriction.

RR-0342: *Do not implement requests that will break generic code sharing.* This request is considered under Requirement R4.4-C(1). This RR discusses how changes in the rules for treatment of static types and expressions could impair generic code sharing possibilities. The RR discusses the potential effect of RR-0027 and RR-0048.

RR-0343: *Provide better facilities for conditional compilation.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0344: *Need to simplify/relax the conformance rules.* This request is considered under Requirement R2.2-B(1).

RR-0345: *Need standardized interface to other ANSI languages.* Rejected: Since interfaces to other programming languages depend on both the language and the implementation, it isn't clear that anything useful can be done to solve the RR's problem in Ada 9X, despite the example solutions given in the RR. See Section 13.1.

RR-0346: *Need portable way to extract mantissa/exponent from floating point number.* This request is considered under Requirement R11.1-A(1).

RR-0347: *Allow applications to change priorities under program control; allow task priority to increase as a function of lack of service.* This request is considered under Requirement R5.2-A(1).

RR-0348: *Need predefined functions for real numbers, e.g., trig, log, etc.* This request is considered under Requirement R11.1-A(1).

RR-0349: *Interrupt addresses and memory addresses are conceptually different and should not be treated the same by the language.* This request is considered under Requirement R6.3-A(2). This RR presents what is believed to be a potential problem, but does not give any specific example of a difficulty imposed by the current approach.

RR-0350: *Clarify wording dealing with default initial values.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document.

RR-0351: *Trusted systems require auto-scrubbing of memory when done with it.* Rejected: not a language issue. See Section 13.7.

RR-0352: *Require Calendar.Clock to return consistently accurate local system time.* This request is considered under Requirement R5.1-B(1). A real-time annex should specify constraints on timing accuracy.

RR-0353: *Unchecked conversion should eliminate compiler-dependent fields.* This request is considered under Requirement R2.4-A(1). The RR points out an important problem in dealing with unchecked conversion, although the proposed solution is not necessarily the correct one.

RR-0354: *Introduce dimensional mathematics into the language.* Rejected: too great a change from Ada 83 (dimensional mathematics). See Section 13.6.

RR-0355: *Standardize means of getting the OS command line arguments.* This request is considered under Requirement R2.4-A(1). At the very least, compilers running under the same operating system should have the same way of interacting with command line arguments. This RR makes an interesting proposal on how to achieve this effect.

RR-0356: *Need a way to get the compilation date and time within a program.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4. This problem can be solved with a suitable environment tool.

RR-0357: *Need packed decimal, wide-ranging fixed-point, decimal deltas.* This request is considered under Study Topic S10.1-A(2).

A: Numerical Listing of RRs

RR-0358: *Need support for floor, ceiling, truncate, and whole operations.* This request is considered under Requirement R11.1-A(1).

RR-0359: *Allow mixed case output for enumeration literals.* This request is considered under Requirement R4.6-B(1).

RR-0360: *Add picture-formatting capabilities to TEXT_IO.* This request is considered under Study Topic S10.4-A(2). The requirement does not go this far, but does suggest adding picture-formatting functions in a separate package.

RR-0361: *Increase the number of options for controlling the output format of numbers.* This request is considered under Requirement R4.6-B(1).

RR-0362: *Allow optional when_clause on the raise statement.* This request is considered under Section A.3.4.

RR-0363: *Allow 'VALUE and 'IMAGE to apply to real types as well as discrete types.* This request is considered under Section A.3.1.

RR-0364: *Allow a subprogram body to be defined by generic instantiation.* This request is considered under Section A.4.1. An example is given using an instantiation of UNCHECKED_CONVERSION.

RR-0365: *Reduce allowed variations in implementations to increase portability.* This request is considered under Requirement R2.4-A(1). RR-0432 is an expanded version of this RR.

RR-0366: *Subtype natural should not include zero.* Rejected: too great a change from Ada 83. See Section 13.6. This change would be dangerously non-upward compatible. Moreover, mathematicians disagree on this.

RR-0367: *Need support for national language character sets, including string comparison.* This request is considered under Requirement R3.1-A(1). The request for string comparison operations was not accepted, for reasons given in the Requirements document in the discussion following the requirement.

RR-0368
This RR concerns two topics, each of which is treated separately. They are now listed.

RR-0368A: *Ensure unnecessary recompilation is avoided.* This request is considered under Study Topic S4.3-A(1).

RR-0368B: *Ensure the library can be manipulated by tools other than those provided by the compiler vendor.* This request is considered under Study Topic S4.3-C(1).

RR-0369: *Provide support for floating point standard IEEE-754.* This request is considered under Study Topic S11.1-B(1).

RR-0370
This RR concerns five topics, each of which is treated separately. They are now listed.

RR-0370A: *Can't recover space declared in library units when reconfiguring a system.* This request is considered under Requirement R8.2-A(1).

RR-0370B: *Can't restart library level tasks.* This request is considered under Requirement R8.2-A(1).

RR-0370C: *Library level tasks can't terminate.* This request is considered under Requirement R2.1-A(1). AI-00399 explains when such tasks can terminate.

RR-0370D: *Need to set priorities of tasks during mode shifts.* This request is considered under Requirement R5.2-A(1).

RR-0370E: *Need to recover space for task control blocks when tasks are created by an allocator.* This request is considered under Requirement R4.2-A(1).

RR-0371: *Need more usable and portable machine code insertions.* Rejected: not desirable to ease machine code insertion. See Section 13.1.1.

RR-0372: *Solve problem where heterogeneous processors view memory differently.* Rejected: a contradictory requirement was made. See Section 13.2. Dealing with heterogeneous shared memory systems is beyond the scope of the requirements (see Requirement R8.1-A(1)).

RR-0373: *Need to be able to dynamically alter a program as it is running.* This request is considered under Requirement R8.2-A(1).

RR-0374: *Ada should address memory management requirements in distributed systems.* This request is considered under Requirement R4.2-A(1).

RR-0375: *Include formal memory protection/security.* Rejected: not a language issue. See Section 13.7. The ability to restrict access to pages of memory is too operation-system dependent to be a suitable language requirement.

RR-0376: *Need special treatment of exceptions in distributed/parallel/multi-processor systems.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3. The RR is very short and does not clearly indicate what problem needs to be solved.

RR-0377: *Ada should allow partitioning of programs for multiple processor environments.* This request is considered under Requirement R8.2-A(1).

RR-0378: *Need standard means of communication in distributed system.* This request is considered under Requirement R8.1-A(1).

RR-0379: *Application should select the specific scheduling algorithm.* This request is considered under Requirement R5.2-A(1).

RR-0380: *Need a task identifier for every task.* This request is considered under Study Topic S7.2-A(1). The RR gives a lengthy discussion of possible uses of task identifiers.

RR-0381: *Records should have composed operations with respect to components.* This request is considered under Requirement R2.2-C(1).

RR-0382: *Need to be able to rename and append to a file in standard Ada.* This request is considered under Requirement R4.6-B(1).

RR-0383: *Need generic exceptions for truly reusable generic units.* This request is considered under Study Topic S4.4-A(1).

RR-0384: *Cannot write subprogram which causes an exception after specified delay.* This request is considered under Requirement R5.1-C(1).

RR-0385: *Need finalization code for packages.* This request is considered under Study Topic S4.2-A(2).

RR-0386: *Need standard way of telling the compiler not to optimize.* This request is considered under Requirement R9.1-A(2).

RR-0387: *Relax 11.6 optimization rules to allow compiler to do more optimizing.* This request is considered under Requirement R2.2-A(1).

RR-0388: *Proposal for clean way of executing a subprogram by its address.* This request is considered under Requirement R4.1-B(1). A straightforward subprogram type provides a simpler solution than the approach proposed in this RR.

RR-0389: *There is a need for ''cyclic'' discrete types in the language.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0390: *Need 8-bit unsigned CHARACTER for Greek and graphics symbols.* This request is considered under Requirement R3.1-A(1).

RR-0391: *Clumsy syntax for based numbers, especially in aggregates.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0392: *Need ''semi-limited'' type with predefined := but no predefined =.* Rejected: too much implementor change for the payoff. See Section 13.5.

RR-0393: *Can't get direct visibility of fixed point mult and div operator by renaming.* This request is considered under Section A.2.3.

RR-0394: *Merge concepts of task and package into concept of an object.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3. Insufficient motivation is given for the requested change.

RR-0395: *Include formal parameter names in parameter/result-type profile.* This request is considered under Requirement R2.2-B(1). The RR points out that it is illegal to declare two subprograms with the same parameter and result type profile in the same declarative region even if corresponding formal parameter names are different. This illegality seems inconsistent to programmers since such subprograms can be unambiguously called using named parameter associations, and moreover, such overloadings can occur as a result of USE clauses and generic instantiations (see, e.g., 12.3(22)). The Mapping/Revision Team may wish to consider whether this apparent irregularity should be preserved in Ada 9X. The RR points out that allowing such declarations may cause problems with renaming declarations, since if the subprogram being renamed is overloaded in this way, the overloading cannot be disambiguated based on the formal parameter names of the renamed subprogram.

RR-0396: *Add library unit elaboration ordering rules to reduce need for pragma ELABORATE.* This request is considered under Section A.2.1.

RR-0397: *Replace keyword PRAGMA with something capturing meaning better.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0398: *Need clearer/more selective rules for pragma INLINE applicability.* This request makes a useful suggestion for improvement in controlling the effect of pragma INLINE. See Section 2.2.9 of this document.

RR-0399: *Break up overly broad predefined exceptions, e.g., CONSTRAINT_ERROR.* Rejected: This issue was given thorough consideration in the original design, and insufficient evidence is given in this RR to justify reconsidering the decision. See Section 13.1.

RR-0400: *Do not allow a task to die silently on an unhandled exception.* This request is considered under Requirement R2.3-A(2). It is not clear what can be done, but the RR does point out a problem.

RR-0401: *Mixed-base fixed-point operations cannot be done efficiently because of accuracy requirements.* This request is considered under Requirement R2.2-A(1). RR-592 duplicates the content of this RR.

RR-0402: *Need unique hierarchical pathnames for subunit.* This request is considered under Study Topic S4.3-C(1).

RR-0403: *Need to be able to get the name of the current exception.* This request is considered under Requirement R4.5-A(1).

RR-0404: *Need convenient way to find out if a particular file exists.* This request is considered under Requirement R4.6-B(1).

RR-0405: *Need convenient way to append to a file.* This request is considered under Requirement R4.6-B(1).

RR-0406: *Allow user-defined attributes for user-defined types.* Rejected: insufficient user benefit (user-defined attributes). See Section 13.4.1.

RR-0407
This RR concerns two topics, each of which is treated separately. They are now listed.

RR-0407A: *Need exception name, line number, and unit name where raised.* This request is consid-

ered under Requirement R4.5-A(1). This RR cites a requirement for logging exception information in information systems. The 9X requirement, however, does not go so far as to request contextual information such as the name of the compilation unit, source code line, etc. The requirement allows additional information to be made available if this can be done with little implementation cost.

RR-0407B: *Do not allow a task to die silently on an unhandled exception.* This request is considered under Requirement R2.3-A(2). It is not clear what can be done, but the RR does point out a problem.

RR-0408: *There is a need for generic formal entries.* This request is considered under Study Topic S4.4-A(1).

RR-0409: *Define in the language how 3.5 rounds to integer.* This request is considered under Requirement R2.4-A(1).

RR-0410: *Provide explicit language support for periodic tasks.* This request is considered under Requirement R5.1-B(1). This RR goes beyond the requirement since it requests direct language support for specifying task periodicity. The arguments should be considered, however, in evaluating Ada 9X proposals.

RR-0411: *Express record representation clauses in a machine-independent way.* This request is considered under Requirement R2.4-A(1).

RR-0412: *Allow overloaded = for all types, not just limited types.* This request is considered under Section A.3.9.

RR-0413: *Allow user-written := for all types.* This request is considered under Study Topic S4.2-A(2).

RR-0414: *Ada needs subprogram types and subprogram objects.* This request is considered under Requirement R4.1-B(1).

RR-0415: *Allow priority inheritance, prioritized entry-queues, and prioritized selective wait.* This request is considered under Requirement R5.2-A(1).

RR-0416: *Granularity of predefined exceptions is too coarse.* Rejected: insufficient user benefit (grouping exceptions). See Section 4.5-A(1). This issue was given thorough consideration in the original design, and insufficient evidence is given in this RR to justify reconsidering the decision.

RR-0417: *Length clause should force allocation of EXACT number of bits.* This request is considered under Requirement R6.2-A(1). The interpretation of length clauses is actively under review by the ARG. In particular, see AI-00536, AI-00553, AI-00561, and AI-00825.

RR-0418: *Representation clauses for array types need to be added.* This request is considered under Requirement R2.2-C(1).

RR-0419: *Add some form of support for varying length strings to the language.* This request is considered under Study Topic S10.4-A(1).

RR-0420: *Need file ''extend'' or ''append'' capability* . This request is considered under Requirement R4.6-B(1).

RR-0421
This RR concerns four topics, each of which is treated separately. They are now listed.

RR-0421A: *Need to delay in processing an interrupt.* This request is considered under Requirement R6.3-A(1).

RR-0421B: *Interrupt address structure is sometimes different from memory address structure; a single type for both is inappropriate.* This request is considered under Requirement R6.3-A(2). No specific examples of problems are given.

RR-0421C: *Need to associate interrupts with entries of task objects, not task types.* This request is considered under Requirement R6.3-A(2).

RR-0421D: *The treatment of interrupts as ordinary, timed, or conditional calls may depend inappropriately on the run-time system.* This request is considered under Requirement R6.3-A(1). The point here is that the run-time system may insulate the application program too completely from hardware-dependent behavior, and so different implementations may behave differently even for the same target hardware.

RR-0422: *Allow subprograms as parameters and maybe also as values.* This request is considered under Requirement R4.1-B(1).

RR-0423: *Remove discriminant restriction on full declarations of private types.* This request is considered under Requirement R2.2-C(1). The RR raises some points worthy of consideration. It refers to AI-00037 for a complete discussion of the problem.

RR-0424: *Allow names exported from an instance to be redefined during instantiation.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0425: *Need open ranges in declarations of real subtypes.* Rejected: There is no obvious notation, and the change is of marginal benefit. See Section 13.1.

RR-0426
This RR concerns four topics, each of which is treated separately. They are now listed.

RR-0426A: *The effect of an optional package body is confusing to users.* This request is considered under Section A.2.4.

RR-0426B: *Allow declaration and body to be combined for generic subprograms.* This request is considered under Requirement R2.2-B(1).

RR-0426C: *Omitting index constraint in constant arrays causes programmer errors.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0426D: *Optional index in 'FIRST (and others) causes problems.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0427: *Do not permit a function to return a locally-declared task object.* This request is considered under Section A.1.1.

RR-0428: *Order of declarations is too restrictive.* This request is considered under Section A.2.2. Specific anomalies mentioned are the inability to specify an address clause immediately after an entry declaration and the inability to specify a representation clause after a body has been declared.

RR-0429: *Need construct that makes just overloadable declarations directly visible.* This request is considered under Section A.2.3.

RR-0430
This RR concerns two topics, each of which is treated separately. They are now listed.

RR-0430A: *Need objects of a subprogram ''type''.* This request is considered under Study Topic S4.1-A(1).

RR-0430B: *Need to pass subprograms as parameters.* This request is considered under Requirement R4.1-B(1).

RR-0431: *A terminate alternative cannot be used to stop cyclic tasks.* This request is considered under Requirement R5.3-A(1). An asynchronous transfer of control construct might serve to meet the need described here.

RR-0432: *Severely limit implementation options to improve portability.* This request is considered under Requirement R2.4-A(1). This RR gives a very extensive list of sections in the Standard that allow implementation-dependent choices to be made.

RR-0433: *There is a need for predefined unsigned integer types.* This request is considered under Requirement R6.1-A(1).

RR-0434: *Need atomic read/write operations on shared volatile memory .* This request is considered under Requirement R7.1-A(1).

RR-0435: *Need secondary standard for simple Ada subset for safety-critical applications.* This request is considered under Requirement R9.3-A(1). The requirement does not propose that Ada 9X will provide such a standard, but it does allow an independently-developed standard to be enforced.

RR-0436: *Clarify task synchronization point inconsistencies.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document. This problem should be addressed.

RR-0437: *Provide ''supertype'' capability for merging enumeration types.* Rejected: insufficient user benefit (discontiguous subtypes). See Section 13.5.3.

RR-0438: *Allow use of multi-octet character set.* This request is considered under Requirement R3.1-A(2).

RR-0439: *Require automatic garbage collection.* This request is considered under Requirement R4.2-A(1).

RR-0440: *Extend Ada to be truly object-oriented.* This request is considered under Study Topic S4.3-B(1).

RR-0441: *Extend Ada to allow for polymorphism.* This request is considered under Study Topic S4.1-A(1).

RR-0442: *Extend Ada to allow a package type hierarchy.* This request is considered under Study Topic S4.3-B(1).

RR-0443: *Need for anonymous array types as record components.* Rejected: insufficient user benefit (anonymous arrays as record components). See Section 13.4.4. LSN-222 discusses the potential complexity of allowing this capability. See Language Study Notes, 1983, available from the Ada Information Clearinghouse.

RR-0444: *Let the user limit the places where a given exception can be raised.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0445: *Non-staticness of generic formals poses problems.* This request is considered under Study Topic S4.4-A(1). This RR argues that it should generally be possible to turn a non-generic unit into a generic unit, but this is not easily possible when the non-generic unit uses static expressions (in case statements, aggregates, and type declarations) that depend on formal parameters in the generic version.

RR-0446: *Tighten the contract model by distinguishing constrained/unconstrained generic types.* This request is considered under Study Topic S4.4-B(2).

RR-0447: *Need to be able to preserve/restore the default file at any point.* This request is considered under Requirement R4.6-B(1). This RR provides a useful argument for the requested capability.

RR-0448: *Allow different sets of subprograms to depend on common declarations.* This request is considered under Study Topic S4.3-B(1).

RR-0449: *Do not allow unchecked conversion of private types.* Rejected: Unchecked conversion exists as an escape mechanism whose usage should not be restricted by the language. Local controls on its use could be enforced in response to Requirement R9.3-A(1). See Section 13.1.

RR-0450: *Need efficient manipulation of buffers whose type is determined at run time.* This request is considered under Study Topic S6.4-B(1). The RR gives an example of a use of the capabilities called for in the requirement.

RR-0451: *Changes to package constants should not cause recompilation.* This request is considered under Study Topic S4.3-A(1).

RR-0452: *Allow constant functions in static expressions (or overloadable constants).* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4. The RR gives no examples showing why such functions are needed in contexts where the language requires static expressions (e.g., in the choice of a case statement).

RR-0453: *Provide a special function or attribute yielding the sign of a numeric value.* This request is considered under Requirement R11.1-A(1).

RR-0454: *Need Entier function or attribute for real types.* This request is considered under Requirement R11.1-A(1).

RR-0455: *The import and export mechanisms of Ada are too limited* . This request is considered under Study Topic S4.3-B(1). This RR contains a fairly extensive discussion of problems that are relevant to the Study Topic.

RR-0456: *Allow initialization to be associated with a type definition.* This request makes a useful suggestion for improvement in default initialization capabilities. See Section 2.2.2 of this document.

RR-0457: *Structure library units as groups, control visibility of library units.* This request is considered under Study Topic S4.3-C(1).

RR-0458: *Need convenient way to escape into weakly typed subprogram call.* This request is considered under Study Topic S4.4-A(1).

RR-0459: *Improve support for interoperability; lessen implementation dependence.* This request is considered under Requirement R2.4-A(1). This RR lists several areas for consideration: representation clauses, the effect of pragma PACK, the effect of unchecked conversion, and permissible optimizations. An extensive and helpful discussion is provided.

RR-0460: *Ada needs to provide support for unsigned integer types.* This request is considered under Requirement R6.1-A(1). This RR provides an extensive discussion of the issues and a detailed solution that helps to indicate the full range of the requirement.

RR-0461: *Provide standard package of semaphore operations.* This request is considered under Requirement R5.2-A(2).

RR-0462: *Allow selected component form of type mark in a formal part even when the selected component has the same identifier as the subprogram.* This request is considered under Section A.3.7.

RR-0463: *'Size is unclear; perhaps need 'Spacing and 'Allocation.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0464: *Should be able to set STORAGE_SIZE for task objects as well as types.* This request is considered under Section A.3.5.

RR-0465: *Need a way to get the representation from an enumeration value and vice versa.* This request makes a useful suggestion for improvement in the ability to determine the representation of enumeration values. See Section 2.2.14 of this document.

RR-0466: *Allow user-defined finalization for objects of a type to ensure release of resources.* This request is considered under Study Topic S4.2-A(2).

RR-0467: *Need convenient way to rename a type and get its operations.* This request is considered under Section A.2.3.

RR-0468: *No generic way to handle exceptions raised by generic formal subprograms.* This request is considered under Study Topic S4.4-A(1).

RR-0469: *Parameter names for language-defined pragmas should be defined.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0470: *Allow renaming or generic instantiation to define a subprogram body.* This request is considered under Section A.4.1. Reasonable examples are given.

RR-0471: *Allow specification of parameter modes in subprogram calls for clarity.* Rejected: too great a change from Ada 83. See Section 13.6. This feature was considered explicitly and rejected in the initial design.

RR-0472: *Distinguish unconstrained/constrained generic formal types.* This request is considered under Study Topic S4.4-B(2).

RR-0473: *Allow ''partially'' constrained subtypes of discriminated records.* Rejected: too much implementor change for the payoff. See Section 13.5.

RR-0474: *Need direct visibility to just enumeration literals and operators of a type.* This request is considered under Section A.2.3.

RR-0475: *Need automatically-invoked user-defined routines to reclaim storage.* This request is considered under Study Topic S4.2-A(2).

RR-0476: *Allow user-written type-conversion functions with the same name as the target type.* Rejected: too great a change from Ada 83. See Section 13.6. The proposal would require significant change to the visibility rules and the overload resolution model.

RR-0477: *Provide a way to get the name and location of a raised exception.* This request is considered under Requirement R4.5-A(1). Obtaining traceback information is not part of the 9X requirement.

RR-0478: *Add language facilities for restricting use of resources to trusted packages.* Rejected: Pro-

viding special-purposes pragmas for such purposes is beyond the scope of the revision effort. See Section 13.1.

RR-0479: *Need standard subprograms to get user-interface information from OS.* Rejected: This is not a bad idea, but there are more important issues that deserve attention. See Section 13.1.

RR-0480: *Need standard means of sending messages between Ada programs.* This request is considered under Requirement R8.1-A(1).

RR-0481: *Make Ada documentation available in SGML format.* This request is considered under Requirement R2.1-C(1).

RR-0482: *Multiple derived types from same package do not generate needed operations.* This request is considered under Study Topic S4.3-B(1). Specialization/extension facilities should help solve this problem.

RR-0483: *Allow an instantiated subprogram to have the same identifier as the generic unit (as is allowed for package instances).* This request is considered under Section A.3.7.

RR-0484: *Add DEFAULT_xy functionality as parameters to generic TEXT_IO packages.* This request is considered under Requirement R4.6-B(1). This RR proposes a simple, upward-compatible solution that improves the usability of the numeric IO packages.

RR-0485: *Provide means to get the line length of an input or output device.* This request is considered under Requirement R4.6-B(1).

RR-0486: *Allow generic formal task types as well as generic formal limited types.* This request is considered under Study Topic S4.4-A(1). This RR points out that if a generic unit expects a task as an actual parameter, the programmer is unable to express this requirement.

RR-0487: *Need private task entries for exclusive use within the task.* This request makes a useful suggestion for improvement in restricting the visibility of task entries. See Section 2.2.11 of this document.

RR-0488: *Allow generic formal entries as well as generic formal subprograms.* This request is considered under Study Topic S4.4-A(1). This RR gives an example of how a generic formal entry would be used to achieve the effect of an asynchronous call.

RR-0489: *Allow machine-code insertions in functions as well as procedures.* Rejected: not desirable to ease machine code insertion. See Section 13.1.1.

RR-0490: *Need successful/convenient recovery from exceptions in machine code insertions.* This request is considered under Requirement R2.3-A(2). This simple request might improve safety of use of machine code insertions.

RR-0491: *Code would be clearer if one could EXIT from a block statement.* Rejected: too difficult to distinguish exiting from a block inside a loop. See Section 13.1.2.

RR-0492: *Decouple mantissa and exponent information in floating point type definitions.* This request is considered under Study Topic S11.1-B(1).

RR-0493: *A programmer should be able to ensure that storage will be reclaimed.* This request is considered under Requirement R4.2-A(1).

RR-0494: *Allow slices for any dimension in multidimensional arrays.* Rejected: insufficient user benefit (multi-dimensional slices). See Section 13.4.2.

RR-0495: *Remove leading space in the result of the 'IMAGE attribute for integers.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0496: *Clarify termination of tasks whose masters are library units .* This request is considered under Requirement R2.1-A(1). AI-00399 defines the effect of termination on library-level tasks.

RR-0497: *Presence of default discriminants for types used as generic actual can yield a surprising run-time error.* Rejected: not a language issue. See Section 13.7. The problem raised in this RR is really a problem of correctly using the language rather than a language problem, particularly since the proposed solution would allow a constrained access variable to inadvertently designate an incorrectly constrained object.

RR-0498: *Make selective wait symmetrical with respect to accept statements and entry calls.* This request is considered under Requirement R5.2-A(2).

RR-0499: *Like other ''blocks'', allow exception handlers in accept statements.* This request is considered under Section A.3.2.

RR-0500: *More terms should be hyphenated to improve clarity.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document.

RR-0501: *The Standard should be consistent in delimiting section headings.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document.

RR-0502: *The Standard should be consistent in its use of upper and lower cases.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document.

RR-0503: *Provide subprogram types for dispatcher-style programming.* This request is considered under Study Topic S4.1-A(1). This RR was particularly useful in formulating the associated User Need and requirement.

RR-0504: *Add an exchange operator.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

A: Numerical Listing of RRs

RR-0505
This RR concerns two topics, each of which is treated separately. They are now listed.

RR-0505A: *Provide extendable record types.* This request is considered under Study Topic S4.3-B(1).

RR-0505B: *Allow partial match for records as generic parameters.* This request is considered under Study Topic S4.4-A(1).

RR-0506: *Allow initialization to be associated with a type definition.* This request makes a useful suggestion for improvement in default initialization capabilities. See Section 2.2.2 of this document.

RR-0507: *Provide information/control over row-major or column-major ordering.* This request is considered under Study Topic S11.2-A(1). The RR gives a detailed discussion of the inefficiency caused by Ada rules.

RR-0508: *Allow slices for any dimension in multidimensional arrays.* Rejected: insufficient user benefit (multi-dimensional slices). See Section 13.4.2.

RR-0509: *Allow user-defined attributes for user-defined or private types.* Rejected: insufficient user benefit (user-defined attributes). See Section 13.4.1.

RR-0510: *Re-indexing arrays via type conversions.* This RR is discussed in Section 2.2.10 on page 10 of this document.

RR-0511: *Allow use of a base type within a generic unit.* This request is considered under Study Topic S4.4-A(1).

RR-0512: *Provide subprograms as parameters to subprograms.* This request is considered under Requirement R4.1-B(1). This RR gives some examples of the limitations of using generic parameters as a means of getting the effect of passing subprograms as parameters.

RR-0513: *Allow overloading of = for any type, e.g., returning an array type.* This request is considered under Section A.3.9. As the RR points out, there is no strong reason to limit the result type of the equality operators.

RR-0514: *Provide support for simple parallel threads within a program unit.* This request is considered under Study Topic S7.3-A(1). Some of the requested functionality is included in the requirement.

RR-0515: *Need ability to request indivisible update for specific objects, especially in distributed systems.* This request is considered under Study Topic S4.2-A(2). The submitter objects to the need to explicitly program mutual exclusion when making assignments to specific objects, and would like to have the assignment operation imply indivisible update. This capability could be provided by user-defined assignment.

RR-0516: *Provide more support for object-oriented programming.* This request is considered under Study Topic S4.3-B(1).

RR-0517: *Provide syntax to declare program units free from side-effects*. This request is considered under Requirement R9.3-A(1). It is not clear that the benefits are worth the costs in language complexity and compiler checks.

RR-0518: *Provide syntax to declare subprogram pre/post conditions.* Rejected: The desired checks can be written in the existing language in a way that permits the optimizer to take advantage of the checks. See Section 13.1.

RR-0519: *Simplify overload rules for ambiguous/universal expressions.* Rejected: These issues were considered thoroughly in the original design, and it is unlikely that the rules can be improved in general without introducing other anomalies. Of course, the -1..10 case (Section A.3.12) should be fixed, but this is not an overloading resolution anomaly but rather a special case rule. See Section 13.1.

RR-0520: *Language should distinguish ''sequence'' and ''mapping'' arrays.* Rejected: It is far from clear that adding a new type would create a simpler, less easily misused language. See Section 13.1.

RR-0521: *Need more convenient support for use of shared memory among tasks.* This request is considered under Requirement R5.2-A(2).

RR-0522: *Allow non-discrete record discriminants.* This request is considered under Requirement R2.2-C(1).

RR-0523: *Allow user-defined finalization for objects of a type to ensure release of resources.* This request is considered under Study Topic S4.2-A(2).

RR-0524: *Allow functions to return references to components of objects; allow programmer to ensure pass by reference for any object.* This request is considered under Requirement R6.4-A(1).

RR-0525: *Extend Ada to allow for polymorphism and inheritance.* This request is considered under Study Topic S4.3-B(1).

RR-0526
This RR concerns three topics, each of which is treated separately. They are now listed.

RR-0526A: *Allow exceptions to be grouped under a single name.* Rejected: insufficient user benefit (grouping exceptions). See Section 4.5-A(1).

RR-0526B: *Need to pass exceptions as parameters to generic units and subprograms.* This request is considered under Study Topic S4.4-A(1).

RR-0526C: *Need to determine the name of a raised exception.* This request is considered under Requirement R4.5-A(1).

RR-0527: *Standardize information/conventions used for pragma INTERFACE.* This request is considered under Requirement R4.1-B(2).

RR-0528: *Change Ada character names to recognized names for verbal communication.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document. The problem addressed in the RR is the names assigned to characters in Section 2.1(15) and 2.2(10). The RR cites federal law requiring facilities for the handicapped.

RR-0529: *Allow selection of operations based on run-time queries about properties of types.* Rejected: too much implementor change for the payoff. See Section 13.5. A fully general ability to query type descriptors at run-time is requested here.

RR-0530: *Insufficient support for mutants of limited types.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3.

RR-0531: *Variants of a type can't be usefully supported with current variant record approach.* This request is considered under Study Topic S4.3-B(1). Although this RR discusses variant record limitations, the real problem being addressed is the ability to construct efficiently representable alternative representations for a conceptual type. The RR mentions explicitly that object-oriented languages allow this kind of problem to be solved more straightforwardly.

RR-0532: *Allow same-type record components in different variants to share name.* This request makes a useful suggestion for improvement in variant record declarations. See Section 2.2.6 of this document. RR-0707 provides a careful analysis of this problem.

RR-0533: *Mutually recursive types from different packages cannot be done.* This request is considered under Study Topic S4.3-B(1). This is an example of a problem that might be solvable with suitable facilities for specializing/extending packages and types.

RR-0534: *Allow brackets other than ''('', '')'' in aggregates, etc.* This request makes a useful suggestion for improvement in the kinds of parenthesization allowed by the language. See Section 2.2.13 of this document.

RR-0535: *Provide CEILING and FLOOR numeric operators.* This request is considered under Requirement R11.1-A(1).

RR-0536: *Provide MIN and MAX numeric operators.* This request is considered under Requirement R11.1-A(1).

RR-0537: *Separate integer divide and floating divide as in Pascal.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0538: *Create new loop structure which bans the EXIT statement.* This request is considered under Requirement R9.3-A(1). A pragma could be used to forbid use of the exit statement.

RR-0539: *Allow reading of OUT parameters.* This request is considered under Section A.3.10.

RR-0540: *Allow a new package to build on an existing package.* This request is considered under Study Topic S4.3-B(1).

RR-0541: *Allow user-defined :=, =, DESTROY operations to support memory management.* This request is considered under Study Topic S4.2-A(2). This RR gives a very lengthy discussion and examples showing why user-defined assignment and finalization are needed to provide appropriate memory management functionality under user control.

RR-0542: *One way or another allow usage of private type before its completion declaration.* This request makes a useful suggestion for improvement in the ability to use private types before their full declaration. See Section 2.2.5 of this document. The need here may be met indirectly by solutions for User Need U4.3-B. AI-00327 contains more detail about the problem.

RR-0543: *Allow accept statements in subprograms nested inside tasks.* This request makes a useful suggestion for improvement in the ability to modularize code in task bodies. See Section 2.2.7 of this document.

RR-0544: *Need indivisible update on reference counts.* This request is considered under Study Topic S4.2-A(2). This RR briefly discusses the difficulties of maintaining reference counts for data shared among tasks. It may provide an interesting example to use when evaluating Ada 9X solutions.

RR-0545: *Subunits should not have to be at the outermost compilation unit level.* Rejected: The ability to declare a subunit in a nested block would require extra complications in requiring that all enclosing blocks be named. Allowing subunit declarations in nested units but not in blocks would seem to be a non-uniformity, so there is no easy way to provide the requested capability. See Section 13.1.

RR-0546: *It is too difficult to ensure that pragma ELABORATE is used when it is needed.* This request is considered under Section A.2.1. This RR gives some examples of problems involving pragma ELABORATE.

RR-0547: *Like non-generic subprograms, allow merge of specification/body for generic subprograms.* This request is considered under Requirement R2.2-B(1). RR-0760 duplicates the content of this RR.

RR-0548: *Allow convenient syntax for instantiating a nested generic unit.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0549: *Ensure the use of unconstrained actual types is always legal.* This request is considered under Study Topic S4.4-B(2).

RR-0550: *Allow subprogram bodies to be defined by RENAMES or generic instantiation.* This re-

quest is considered under Section A.4.1. RR-0761 duplicates the content of this RR.

RR-0551: *Need assignment capability for TEXT_IO.FILE_TYPE.* This request is considered under Requirement R4.6-B(1). See RR-0447 for a workaround that can be used today. RR-0762 duplicates the content of this RR.

RR-0552: *Need ''padded'' line input with truncation and pad-fill to 'LENGTH.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0553: *GET_LINE should not automatically call SKIP_LINE.* This request is considered under Requirement R4.6-B(1).

RR-0554: *Need constraint checks for target of Unchecked_Conversion and I/O input.* This request is considered under Requirement R9.1-A(2). Section 11.6 of Standard allows seemingly redundant constraint checks to be optimized away.

RR-0555: *Need ''selective'' USE clause to get just operators and subprograms of a type.* This request is considered under Section A.2.3.

RR-0556: *Parentheses are used for too many purposes in the language.* This request makes a useful suggestion for improvement in the kinds of parenthesization allowed by the language. See Section 2.2.13 of this document.

RR-0557: *The use of renaming declarations to provide subprogram bodies helps get around the inability to overload subunit names.* This request is considered under Study Topic S4.3-C(1).

RR-0558: *Deriver of type should be able to hide subset of derived operations.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4. The ability to hide some derived operations would add more complexity to the language than is acceptable to most users.

RR-0559: *If allow reading of OUT parameters, initialize OUT access to NULL.* Rejected: too great a change from Ada 83. See Section 13.6. There is no particular advantage in initializing an **out** parameter to null rather than to the value of its actual parameter. Moreover, such a change would be inconsistent with the current rule for components of an access type.

RR-0560: *Need to access a private type's representation in related packages.* This request is considered under Study Topic S4.3-B(1).

RR-0561: *Allow case statement to operate on strings for string processing.* Rejected: too much implementor change for the payoff (non-static case labels). See Section 13.5.2.

RR-0562: *Require separate compilation of generic specifications and bodies.* This request is considered under Requirement R4.4-B(1).

RR-0563: *Need to allow subprogram types and variables.* This request is considered under Study Topic S4.1-A(1).

RR-0564: *Allow implementation freedom to include more mantissa digits in floating point safe numbers.* This request is considered under Study Topic S11.1-B(1).

RR-0565: *'SMALL is unsuitably defined; need for representation clauses inappropriate.* This request is considered under Requirement R2.2-B(1).

RR-0566: *Fixed point model numbers should include the bounds of the type definition.* This request is considered under Requirement R2.2-B(1).

RR-0567: *Allow variable declaration to get constraints from initial value.* This request is considered under Requirement R2.2-C(1).

RR-0568: *Allow non-nested variant parts in record types.* This request is considered under Requirement R2.2-C(1).

RR-0569: *Relax rules separating basic from later declarative items.* This request is considered under Section A.2.2.

RR-0570: *Allow the prefix of a name to denote a renaming of an enclosing construct.* Rejected: AI-00119 discusses the reasons for this restriction. See Section 13.1.

RR-0571
This RR concerns two topics, each of which is treated separately. They are now listed.

RR-0571A: *Allow use of OTHERS choice with named associations when index bounds are determined by context.* This request is considered under Section A.2.5.

RR-0571B: *Clarify the effect when the choice in an aggregate is outside the range of the applicable index constraint.* This request is considered under Requirement R2.1-A(1). AI-00309 deals with this problem.

RR-0572: *Need predefined operators with respect to all predefined integer types.* Rejected: The change would require revision of the overloading rules because X**2 would become ambiguous. See Section 13.1.

RR-0573: *Slide indices of array aggregates for record component initialization and as components of record aggregates.* This request is considered under Section A.3.11.

RR-0574: *Inability to eliminate constraint check for OUT parameters.* This request is considered under Requirement R2.2-A(1). This RR points out a situation in which a redundant constraint check must be performed both inside and outside a procedure.

RR-0575: *Need better (more selective) control over inlining.* This request makes a useful suggestion for improvement in controlling the effect of pragma INLINE. See Section 2.2.9 of this document.

RR-0576: *Allow parameter default expressions to make use of previous IN parameters.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4. This request is harder to satisfy than may appear at first, since it requires, in effect, that actual parameters be evaluated in an order dictated by the order of the formal parameters and is a potential complication for implementers without strong benefit to users.

RR-0577: *Allow deferred constant of composite type having a component of an incompletely declared private type.* This request is considered under Requirement R2.2-C(1).

RR-0578: *Out-mode parameters of limited private types should be allowed.* This request makes a useful suggestion for improvement in the treatment of subprogram parameters. See Section 2.2.3 of this document. This RR gives a good example to show why this restriction should be relaxed to allow good modular programming practices to be supported.

RR-0579: *Allow a type mark of form P.FOO in the formal part of a subprogram named FOO.* This request is considered under Section A.3.7.

RR-0580: *Allow accepts within subprograms/packages nested inside tasks.* This request makes a useful suggestion for improvement in the ability to modularize code in task bodies. See Section 2.2.7 of this document.

RR-0581
This RR concerns three topics, each of which is treated separately. They are now listed.

RR-0581A: *Eliminate need for pragma ELABORATE; pragma NOT_ELABORATE might help.* This request is considered under Section A.2.1. This RR contains some detailed discussion and examples.

RR-0581B: *Clarify the effect of applying pragma ELABORATE to a package that has no body.* This request is considered under Requirement R2.1-A(1). AI-00236 specifies the effect of the pragma in these cases.

RR-0581C: *Allow a pragma ELABORATE for a subunit to mention a package name given in the context clause of a parent library unit.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0581: *Rules specifying the position of pragma ELABORATE are error-prone and unhelpful.* This request is considered under Section A.2.1.

RR-0582: *Provide standard interface for getting additional implementation-dependent info about state when an exception is raised.* This request is considered under Requirement R4.5-A(1). The requirement allows additional information to be made available if this can be done with little implementation cost.

RR-0583: *Delete NUMERIC_ERROR if now subsumed under CONSTRAINT_ERROR.* This request is considered under Requirement R2.1-A(1). AI-00387 recommends that NUMERIC_ERROR be replaced with CONSTRAINT_ERROR.

RR-0584: *Need stricter checking of formal generic subtypes when an instantiation is given.* This request is considered under Study Topic S4.4-B(2). This RR provides a careful discussion of an error-prone aspect of generics, namely, the fact that formal subtypes are sometimes ignored in matching actual parameters.

RR-0585: *Need pragma to specify code-generation strategy for generic instantiation.* This request is considered under Requirement R4.4-C(1).

RR-0586: *Different instantiations of the same generic unit may have to evaluate their actual parameters in different orders.* This request is considered under Requirement R4.4-C(1). The RR asserts that for a stack machine, this causes inefficiencies for shared-code generics.

RR-0587: *Provide for communication between loosely coupled tasks.* This request is considered under Requirement R5.4-A(1).

RR-0588: *Provide a form of USE clause that hides outer homographs.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4. RR-0589 duplicates the content of this RR.

RR-0589: *Need stronger kind of USE for less dependence on containing scope.* This RR duplicates the content of RR-0588; it is not discussed further.

RR-0590: *Need clear, efficient, standard support for mutual exclusion.* This request is considered under Requirement R5.2-A(2). This RR gives a detailed example of a problem that is to be solved by improved mechanisms in Ada 9X.

RR-0591: *Allow fixed-point multiply/divide with universal real operands.* This request makes a useful suggestion for improvement in the ability to use real literals in fixed point expressions. See Section 2.2.8 of this document.

RR-0592: *Clean up required accuracy for composite fixed-point operations.* This RR duplicates the content of RR-0401; it is not discussed further.

RR-0593: *Mandate implementation of variant record I/O in DIRECT_IO/SEQUENTIAL_IO.* This request is considered under Requirement R4.6-B(1). The required functionality is available in principle by using ''shared files'', as suggested in the discussion following Requirement R4.6-B(1).

RR-0594: *Relax rules separating basic from later declarative items.* This request is considered under Section A.2.2.

RR-0595: *Allow default initialization for all types.* This request makes a useful suggestion for improvement in default initialization capabilities. See Section 2.2.2 of this document.

A: Numerical Listing of RRs

RR-0596: *Allow END type_name to substitute for END RECORD.* This request is considered under Section A.3.13.

RR-0597: *Need functional version of GET_LINE instead of procedural.* This request is considered under Requirement R4.6-A(1). RR-0047 gives a stronger justification for this change.

RR-0598: *Permit function parameters to have modes OUT and IN OUT.* Rejected: insufficient user benefit (OUT modes for functions). See Section 13.4.3.

RR-0599: *Certain changes to derived/private types will help inheritance.* This request is considered under Study Topic S4.3-B(1). This RR gives some generic examples of difficulties of extending existing units to meet new needs.

RR-0600: *Allow formal parameter names in parameter/result-type profile.* This request is considered under Requirement R2.2-B(1). See the comment for RR-0395.

RR-0601: *Allow library-level declarations to be defined by RENAMES.* This request is considered under Requirement R2.2-C(1).

RR-0602: *Encourage implementors to support standardized libraries.* This request was met — This request is properly met by an Annex. See Section 1.2 of the Requirements Document.

RR-0603: *Allow discontiguous subtypes of discrete types.* Rejected: insufficient user benefit (discontiguous subtypes). See Section 13.5.3.

RR-0604: *Like non-generic subprograms, allow merge of specification/body for generic subprograms.* This request is considered under Requirement R2.2-B(1).

RR-0605: *Rules for OTHERS in aggregates are confusing.* This request is considered under Section A.2.5.

RR-0606: *Allow generic subprogram names to be overloaded.* Rejected: insufficient user benefit (overloaded generic names). See Section 13.5.4.

RR-0607: *Allow names of compilation units to be overloadable, operator symbols.* Rejected: Although it may seem more uniform to allow library unit names to be overloaded, a with clause naming such a unit would be unresolvably ambiguous. See Section 13.1.

RR-0608: *Allow recursive generic instantiations.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3. It is not clear how this could be implemented.

RR-0609: *Allow user-defined override of =, /=, := on all types.* This request is considered under Study Topic S4.2-A(2).

RR-0610: *Why not allow RENAMES for types and subtypes?.* This request is considered under Requirement R2.2-C(1).

RR-0611: *Allow subprogram types, variables, constants, parameters, etc.* This request is considered under Study Topic S4.1-A(1). This RR was helpful in formulating the associated User Need and the requirement.

RR-0612: *Should allow both delay and terminate alternatives in selective wait.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3. The intended semantics is not clear.

RR-0613: *User-defined attributes solve portability problems with implementation-defined attributes.* Rejected: insufficient user benefit (user-defined attributes). See Section 13.4.1.

RR-0614: *Allow WHEN condition RETURN to make selection of returned value clearer.* This request is considered under Section A.3.4.

RR-0615: *Define LOOP/UNTIL control structure as in Pascal.* This request makes a useful suggestion for improvement in iteration constructs. See Section 2.2.12 of this document.

RR-0616: *Require compilers to diagnose statically-detectable constraint errors.* This request is considered under Study Topic S2.3-A(1).

RR-0617: *Eliminate anonymous array types.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0618: *Ban GOTO statement.* Rejected: Tools that produce Ada code need to be able to generate GOTOs. See Section 13.1.

RR-0619: *Eliminate three replacement characters, stick to normal ASCII.* This request is considered under Requirement R2.2-B(1). Simplifying the language by removing these alternatives is not upward compatible, but few programs use these replacement characters.

RR-0620: *Ban RETURN statement except inside functions.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0621
This RR concerns three topics, each of which is treated separately. They are now listed.

RR-0621A: *Need to find out which exception has been raised.* This request is considered under Requirement R4.5-A(1).

RR-0621B: *Permit exceptions as generic formals.* This request is considered under Study Topic S4.4-A(1).

RR-0621C: *Allow case statements to dispatch on value of an exception.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4. The intent here is to call a procedure in an ''others'' handler with the actual exception as a parameter. A case statement in the procedure body dispatches on the actual exception value. This is an interesting idea, but not of sufficient value to fall under one of the stated requirements.

RR-0622: *The Standard should use ''metatype'' in describing generic formal types.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document.

RR-0623: *Define RANGE attribute for discrete ranges.* This request is considered under Section A.3.3.

RR-0624: *Provide selective direct visibility into a package.* This request is considered under Section A.2.3. The requirement addresses some of the request.

RR-0625: *Change EXIT/WHEN to WHEN/EXIT to parallel Ada IF and English.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0626: *Files produced by SEQUENTIAL_IO and DIRECT_IO are not portable among compilers, even for the same target machine e.g., because of dope vectors.* This request is considered under Requirement R6.2-A(1).

RR-0627: *Allow partial match to formal type for records.* This request is considered under Study Topic S4.4-A(1).

RR-0628: *Need private task entries.* This request makes a useful suggestion for improvement in restricting the visibility of task entries. See Section 2.2.11 of this document.

RR-0629: *Need procedure and function types for use in subprogram calls.* This request is considered under Requirement R4.1-B(1).

RR-0630: *Due to high implementation costs, define/allow Ada subsets.* Rejected: a contradictory requirement was made. See Section 13.2. The Requirements Team explicitly considered and rejected the notion of allowing subsets as not being consistent with the goals of the revision effort. See Section 1.2 of the Requirements Document.

RR-0631: *Make conformance rules consistent.* This request is considered under Requirement R2.2-B(1).

RR-0632: *Allow EXIT from a block statement for consistency.* Rejected: too difficult to distinguish exiting from a block inside a loop. See Section 13.1.2.

RR-0633: *Provide logical operations (e.g., XOR) for integers.* This request is considered under Requirement R6.1-A(1).

RR-0634: *Provide arithmetic shift operations for integers.* This request is considered under Requirement R6.1-A(1).

RR-0635: *Provide basic support for extended precision integer arithmetic.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0636: *Improve Ada's axioms for floating point operations.* This request is considered under Study Topic S11.1-B(1).

RR-0637: *Ada programs should run as though negative zero did not exist* . This request is considered under Study Topic S11.1-B(1). This is a complex issue that is being considered by the Numerics Rapporteur Group of ISO-IEC/JTC1/SC22/WG9.

RR-0638: *Axioms for built-in operations should be specified explicitly.* Rejected: This was considered and rejected in the initial design as being unnecessary for clarity and precision. The current wording is adequate. See Section 13.1.

RR-0639: *Need compile-time initialization of complex data structures.* This request is considered under Requirement R8.2-A(1). This problem may be better solved with a separate CASE tool.

RR-0640: *Need to access chunk of a bit vector as a whole.* This request is considered under Requirement R6.1-A(1). The requirement provides much of the requested functionality.

RR-0641: *Add subprograms as parameters to the language.* This request is considered under Requirement R4.1-B(1).

RR-0642: *Add label variables to support use of finite state machines.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0643: *Garbage collection can now be done well; encourage its use.* This request is considered under Requirement R4.2-A(1). Section 4.2 of the Requirements Document addresses this topic further.

RR-0644: *Standard should specify time bounds/constraints for certain operations.* This request is considered under Study Topic S9.1-A(1). This RR provides some possibly helpful examples of performance constraints that might be imposed in an Ada 9X annex.

RR-0645: *Need mantissa/exponent extraction and manipulation.* This request is considered under Requirement R11.1-A(1).

RR-0646: *Allow exceptions to be parameterized with parameters read in handler.* Rejected: a contradictory requirement was made. See Section 13.2. The requirements for exceptions are much less ambitious because of potential implementation overhead.

RR-0647: *Need ability to select actions depending on state without using case statements.* This request is considered under Study Topic S4.1-A(1).

RR-0648: *Need to set STORAGE_SIZE on task objects, not task types.* This request is considered under Section A.3.5.

RR-0649: *Allow default initialization for all types (not just records).* This request makes a useful suggestion for improvement in default initialization capabilities. See Section 2.2.2 of this document.

RR-0650: *Allow non-static case statement choices, non-discrete case statement expression.* Rejected: too much implementor change for the payoff (non-static case labels). See Section 13.5.2.

RR-0651: *Allow one task to raise an exception in another task.* This request is considered under

Requirement R5.3-A(1). A different solution to the problem is called for.

RR-0652: *Declaring a subtype should make the equality operator directly visible.* This request is considered under Section A.2.3.

RR-0653: *Need to declare constants whose value is supplied after linking.* This request is considered under Requirement R8.2-A(1).

RR-0654: *Need non-static priorities.* This request is considered under Requirement R5.2-A(1). This RR notes, in effect, that the appropriate priority of a task depends on what other tasks are executing on the same processor, and this might change during system development, maintenance, or run-time configuration, in which case, the language is too restrictive in requiring that priorities be specified with static expressions.

RR-0655: *Add asynchronous message queues.* This request is considered under Requirement R5.4-A(1).

RR-0656: *Need timed exceptions for deadline scheduling.* This request is considered under Requirement R5.1-C(1). The requirement should provide the requested functionality.

RR-0657: *Order entry queues based on priority.* This request is considered under Requirement R5.2-A(1).

RR-0658: *Allow accept statement possibility in a conditional entry call.* This request is considered under Requirement R5.2-A(2).

RR-0659: *Need to make entry call on a generic formal parameter.* This request is considered under Study Topic S4.4-A(1).

RR-0660: *Need constructors and destructors for package types.* This request is considered under Study Topic S4.2-A(2).

RR-0661: *Need language features for assigning tasks to nodes.* This request is considered under Requirement R8.2-A(1).

RR-0662: *Need package classes and inheritance for object-oriented programming.* This request is considered under Study Topic S4.3-B(1).

RR-0663: *Allow certain overloading of := and subscripting.* This request is considered under Study Topic S4.2-A(2). User-defined subscripting is not required.

RR-0664: *Need 'IMAGE and 'VALUE attributes for floating-point types.* This request is considered under Section A.3.1.

RR-0665
This RR concerns three topics, each of which is treated separately. They are now listed.

RR-0665A: *Support multicast message transfer.* This request is considered under Study Topic S5.4-B(1).

RR-0665B: *Support allocation of parallel processes to processors.* This request is considered under Requirement R8.2-A(1).

RR-0665C: *Support message-driven intertask communication.* This request is considered under Requirement R5.4-A(1).

RR-0666: *Allow a subprogram body to be given by generic instantiation.* This request is considered under Section A.4.1.

RR-0667: *Allow a subprogram body to be given by RENAMES.* This request is considered under Section A.4.1.

RR-0668: *Need package types to get, for example, an array of packages.* This request is considered under Study Topic S4.3-B(1). The requirement provides much of the requested functionality.

RR-0669: *Allow user-written := routines.* This request is considered under Study Topic S4.2-A(2).

RR-0670: *Decouple = and /=; do not distinguish private from limited private.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0671: *Allow exceptions as generic parameters.* This request is considered under Study Topic S4.4-A(1).

RR-0672: *Need anonymous pointer types.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0673: *Allow ''END RECORD type_name'' to substitute for ''END RECORD'').* This request is considered under Section A.3.13.

RR-0674: *Allow user-defined attributes as functions.* Rejected: insufficient user benefit (user-defined attributes). See Section 13.4.1.

RR-0675: *Allow a subprogram identifier to be used as a type mark in its specification.* This request is considered under Section A.3.7.

RR-0676: *Add finalization to ensure release of resources.* This request is considered under Study Topic S4.2-A(2).

RR-0677: *Allow initialization clauses on scalar type declarations.* This request makes a useful suggestion for improvement in default initialization capabilities. See Section 2.2.2 of this document.

RR-0678: *Pragma SHARED is not sufficient for data shared between programs; need VOLATILE.* This request is considered under Requirement R7.1-A(1).

RR-0679: *Allow component selection on objects of a private type.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0680: *Predefined exponentiation should take any integer type for exponent.* Rejected: This change is difficult to make because of the overload resolution rules. This problem was considered extensively in the initial design, and all solutions posed difficulties to users. There are more important changes to focus on in this revision of Ada. See Section 13.1.

RR-0681: *A definition of an Ada Line Of Code (LOC) should be standardized.* Rejected: not a language issue. See Section 13.7.

RR-0682: *Allow user-defined overloaded operators such as ''?'', '':-'', etc.* Rejected: too much implementor change for the payoff (user-defined operator syntax). See Section 13.5.1.

RR-0683: *Section 11.6 of the Standard is unclear about what replacements are allowed.* This request is considered under Requirement R2.2-A(1).

RR-0684: *Related packages need access to a private type's representation.* This request is considered under Study Topic S4.3-B(1).

RR-0685: *Clarify and loosen 11.6 to allow more optimization.* This request is considered under Requirement R2.2-A(1).

RR-0686: *Priority of interrupts higher than normal tasks is ill-conceived.* This request is considered under Requirement R6.3-A(1).

RR-0687: *Pragma INLINE should not apply to all overloads; only closest.* This request makes a useful suggestion for improvement in controlling the effect of pragma INLINE. See Section 2.2.9 of this document.

RR-0688: *Unnecessary recompilation required when redeclaring a subprogram body.* This request is considered under Study Topic S4.3-A(1). This RR gives an example where recompilation should not be required.

RR-0689: *Optional bodies should not be unlinked without a warning.* This request is considered under Section A.2.4.

RR-0690: *Allow incomplete and private types to be completed by subtype declaration.* This request is considered under Section A.4.2.

RR-0691: *Allow machine-code insertions in functions as well as procedures.* Rejected: not desirable to ease machine code insertion. See Section 13.1.1.

RR-0692: *Allow implementation-defined pragmas to cause unsuccessful compilation if restrictions implied by the pragmas are not obeyed.* This request is considered under Study Topic S2.3-A(1).

RR-0693: *Parameter passing rules for scalars makes generic code sharing hard.* This request is considered under Requirement R2.2-A(1).

RR-0694: *Need easy direct visibility to the equality operations.* This request is considered under Section A.2.3.

RR-0695: *Allow EXIT from block for legibility.* Rejected: too difficult to distinguish exiting from a block inside a loop. See Section 13.1.2.

RR-0696: *Pragmas LIST and PAGE should be optional.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0697: *Allow entry call alternative in selective wait.* This request is considered under Requirement R5.2-A(2). RR-0498 provides a rationale for this capability.

RR-0698: *Need ability to separate portable and non-portable code into separate units.* This request is considered under Study Topic S4.3-B(1).

RR-0699: *Do not treat an unaccepted length clause for a type as an error.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3.

RR-0700: *Ensure that constant functions like sin(10.0) are evaluated at compile-time.* Rejected: It is too difficult to define what functions should be evaluated at compile-time. Moreover, the change would pose the potentially severe implementation burden of requiring a target machine function to be evaluated in the host machine environment. See Section 13.1.

RR-0701: *Allow specification of STANDARD in the same way as for SYSTEM.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

RR-0702: *There is a need for improvements in heap storage management.* This request is considered under Requirement R4.2-A(1).

RR-0703: *Need to specify STORAGE_SIZE on task objects, not task types.* This request is considered under Section A.3.5.

RR-0704: *Make every bit available to the application programmer.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3.

RR-0705: *For better performance, remove restrictions on static expressions.* This request is considered under Requirement R2.2-A(1). The RR provides a detailed analysis of approved AIs in terms of their possible effect on efficiency of both compilation and execution.

RR-0706: *Allow exceptions and packages as generic parameters.* This request is considered under Study Topic S4.4-A(1).

RR-0707: *Need same-name component identifiers in different variants.* This request makes a useful suggestion for improvement in variant record declarations. See Section 2.2.6 of this document. This RR gives a careful analysis of the unpleasant workarounds required today, but it is not clear that the implementation impact of this change would be acceptable.

RR-0708: *Allow infix function calls.* Rejected: too much implementor change for the payoff. See Section 13.5.

RR-0709: *Need more portability in getting command line inputs.* This request is considered under Requirement R2.4-A(1). See RR-0355 for specific suggestions.

RR-0710: *Need to tie task entries to asynchronous external events generated by operating system.* This request is considered under Requirement R6.3-A(2).

RR-0711: *I/O by a task in multi-task application should not block whole program.* This request is considered under Requirement R4.6-B(1).

A: Numerical Listing of RRs

RR-0712: *Need ability to declare double precision numeric types within a generic unit.* This request is considered under Study Topic S4.4-A(1). The problem here is mainly the inability to create new numeric types whose precision is a function of a generic formal type. Relaxing the rules concerning static expressions would help in the creation of numerical library packages.

RR-0713: *Relax array matching rules for generics.* This request is considered under Study Topic S4.4-A(1).

RR-0714: *Allow default names for all generic formal parameters.* This request is considered under Section A.3.8. The RR gives a detailed example showing the problem caused by the inability to associate default names with generic formal types.

RR-0715: *Allow user-defined type conversions and attributes for numeric types.* This request is considered under Requirement R2.2-C(1). The ability to allow programmers to build user-defined types that have the same attributes and conversion notation as the predefined types is attractive as a generalization of the language's existing capability, but it is unclear whether such changes can be made without introducing anomalies.

RR-0716: *Unify and add attributes for numeric types.* This request is considered under Requirement R11.1-A(1). Many of the requested functions are being provided in packages currently being considered for ISO standardization.

RR-0717: *Allow specification of a step size in FOR loops.* This request makes a useful suggestion for improvement in iteration constructs. See Section 2.2.12 of this document.

RR-0718: *Need predictable results in numeric computation, especially regarding optimization.* This request is considered under Requirement R9.1-A(2). This RR gives an example of how optimization might cause difficulty in evaluating carefully constructed numerical expressions.

RR-0719: *Need standard for trig functions, sqrt, etc.* This request is considered under Requirement R11.1-A(1).

RR-0720: *Floating-point model should reflect actual hardware architectures.* This request is considered under Study Topic S11.1-B(1).

RR-0721: *Try to add unsigned integers to the language.* This request is considered under Requirement R6.1-A(1).

RR-0722: *Need generic formal record types.* This request is considered under Study Topic S4.4-A(1).

RR-0723: *Need support for reconfiguration in emergency cases.* This request is considered under Requirement R8.2-A(1).

RR-0724: *Need clearer/simpler overload resolution rules, especially for implicit conversion.* This request is considered under Requirement R2.1-A(1). The problem mentioned here is addressed by AI-00136 and AI-00606.

RR-0725: *Need rename in package body for routine in package specification.* This request is considered under Section A.4.1.

RR-0726: *Need non-contiguous arrays, static pointers.* This request is considered under Requirement R6.4-A(1). The requirement supplies much of the requested functionality.

RR-0727: *Need selective direct visibility of package declarations.* This request is considered under Section A.2.3. The requirement reflects some of the requested functionality.

RR-0728: *Need simple Ada run-time system for distributed memory MIMD architectures.* This request is considered under Requirement R8.1-A(1). This RR asks for simplifications that reduce the size of the runtime system that must be supported on each node of a distributed system. No specific suggestions are made.

RR-0729: *Language should provide way to turn off optimization to eliminate bugs.* This request is considered under Requirement R9.1-A(2).

RR-0730: *The private part of a package should have its own context clause.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4. Study Topic S4.3-B(1) may help to alleviate this problem.

RR-0731: *Use the Language Compatible Arithmetic Standard as a basis for Ada's floating point model.* This request is considered under Study Topic S11.1-B(1).

RR-0732: *Clarify semantics of instantiating ENUMERATION_IO with an integer type.* This request is considered under Requirement R2.4-A(1).

RR-0733: *Need fixed-point types not centered on zero.* Rejected: too much implementor change for the payoff. See Section 13.5.

RR-0734: *Generalize cases that allow implicit subtype conversion.* This request is considered under Section A.3.11.

RR-0735: *Need ability to change interrupt bindings at run-time.* This request is considered under Requirement R6.3-A(2).

RR-0736: *Need 8-bit ASCII in Ada* . This request is considered under Requirement R3.1-A(1).

RR-0737: *Allow reliable user control over selection of alternatives in a select statement.* This request is considered under Requirement R5.2-A(1). The RR explicitly notes that the 'COUNT attribute is not sufficient to ensure the requested degree of control.

RR-0738: *Add facilities to support vector processing hardware.* This request is considered under Study Topic S7.3-A(1). Although the requirement does not suggest that vector types and operands be added to the language, it does require that the revision address the needs of vector processing hardware.

RR-0739: *Relax 11.6 canonical order rules to allow more optimization.* This request is considered under Requirement R2.2-A(1).

RR-0740: *For optimization with respect to inlined subprograms, allow merging of scopes.* This request is considered under Requirement R2.2-A(1).

RR-0741: *Need hot performance on vector machines; add vector types and operands.* This request is considered under Study Topic S7.3-A(1).

RR-0742: *Need ability to asynchronously stop another task.* This request is considered under Requirement R5.3-A(1).

RR-0743: *Need to allow increment of something other than one in for loops.* This request makes a useful suggestion for improvement in iteration constructs. See Section 2.2.12 of this document.

RR-0744: *Allow for loop to have non-discrete (fixed-point) parameter.* This request makes a useful suggestion for improvement in iteration constructs. See Section 2.2.12 of this document.

RR-0745: *Add facilities for dimensional mathematics to the language.* Rejected: too great a change from Ada 83 (dimensional mathematics). See Section 13.6.

RR-0746: *Allow pictures/graphics as comments in source code.* Rejected: not a language issue. See Section 13.7. The environment should provide this functionality.

RR-0747: *Provide better support for ''light-weight'' parallelism (as in Linda).* Rejected: too great a change from Ada 83. See Section 13.6. Although the LINDA model may be attractive, such a change of concept is outside the scope of the 9X revision effort.

RR-0748: *Provide standard package of asynchronous primitives.* This request is considered under Requirement R5.4-A(1).

RR-0749: *Should allow index sliding for slices serving as actual parameters and as values in record components.* This request is considered under Section A.3.11.

RR-0750: *Add support for inheritance and polymorphism to the language.* This request is considered under Study Topic S4.3-B(1).

RR-0751: *Add WHEN/RAISE construct to the language.* This request is considered under Section A.3.4.

RR-0752: *Make various improvements to exception handling capabilities.* This RR duplicates the content of RR-0621; it is not discussed further.

RR-0753: *Make syntax for task type declarations more consistent.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0754: *Require warnings for unrecognized pragmas.* This request is considered under Study Topic S2.3-A(1).

RR-0755: *Allow ''['' instead of ''('' for indexed components.* This request makes a useful suggestion for improvement in the kinds of parenthesization allowed by the language. See Section 2.2.13 of this document.

RR-0756: *Require warnings when pragmas are ignored.* This request is considered under Study Topic S2.3-A(1).

RR-0757: *Clean up definitions of program unit and compilation unit.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document. This clarification may be worthwhile.

RR-0758: *Bad paragraph numbering.* This request makes a useful suggestion for improvement in clarifying the wording of the standard. See Section 2.1-A(2) of this document.

RR-0759: *Add real-time and verification facilities for control engineering.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3.

RR-0760: *Like non-generic subprograms, allow merge of specification/body for generic ones.* This RR duplicates the content of RR-0547; it is not discussed further.

RR-0761: *Allow subprogram bodies to be defined by RENAMES or generic instantiation.* This RR duplicates the content of RR-0550; it is not discussed further.

RR-0762: *Need assignment capability for TEXT_IO.FILE_TYPE.* This RR duplicates the content of RR-0551; it is not discussed further.

RR-0763: *Allow nested scopes to turn off pragma SUPPRESS.* This request is considered under Requirement R2.3-A(2). Enforcing conventions for the correct use of pragma SUPPRESS can be important. The best solution, however, is not necessarily a pragma that turns off the effect of the pragma in a nested scope.

RR-0764: *Allow subprogram bodies to be defined by RENAMES.* This request is considered under Section A.4.1. This RR argues that the workaround needed when a subprogram body can't be provided by a renaming declaration increases recompilation requirements.

RR-0765: *Allow ''when Package_Name.others =>'' as exception handler.* Rejected: This change could introduce serious problems during maintenance. See Section 13.1.

RR-0766: *Allow bit-wise operations (AND, SHIFT) on integers, bytes, etc.* This request is considered under Requirement R6.1-A(1).

RR-0767: *Solve the elaboration order problem without requiring the use of pragma ELABORATE.* This request is considered under Section A.2.1.

RR-0768: *Need to asynchronously interrupt another task to stop it.* This request is considered under Requirement R5.3-A(1).

A: Numerical Listing of RRs

RR-0769: *Correct wording in the definition of ancestor unit.* This request is considered under Requirement R2.1-A(1). See AI-00482.

RR-0770: *Make aborting yourself cause instant completeness.* Rejected: not a language issue. See Section 13.7. It is arguable that 9.10(6) requires immediate completion of a task that aborts itself.

RR-0771: *Require tasks to have an accept for each entry.* This request is considered under Requirement R9.3-A(1). Requiring at least one accept statement for each entry may be a reasonable project coding convention that should be enforceable by compilers.

RR-0772: *Need to be able to get exception name in a handler.* This request is considered under Requirement R4.5-A(1).

RR-0773: *Need to pack variable-length records into a block for data transmission.* This request is considered under Requirement R6.2-A(1).

RR-0774
This RR concerns many topics, each of which is treated separately. They are now listed.

RR-0774A: *Make it possible to write NEW in Ada.* This request is considered under Requirement R4.2-A(1).

RR-0774B: *Tasking defined as a standard package of functions.* Rejected: too great a change from Ada 83. See Section 13.6.

RR-0774C: *Extend control of library unit visibility.* This request is considered under Study Topic S4.3-C(1).

RR-0774D: *Allow overloaded names in the library.* Rejected: Although it may seem more uniform to allow library unit names to be overloaded, a with clause naming such a unit would be unresolvably ambiguous. See Section 13.1.

RR-0774E: *Provide access to context of an exception situation.* This request is considered under Requirement R4.5-A(1). The requirement allows additional information to be made available if this can be done with little implementation cost.

RR-0774F: *Allow aliased exceptions within the same exception handler.* This request is considered under Requirement R2.2-B(1). It seems that it would be both useful and harmless to allow both P1.END_ERROR and P2.END_ERROR as exception choices in a single exception handler when both exceptions denote IO_EXCEPTIONS.END_ERROR.

RR-0774G: *Provide exception name in OTHERS handler.* This request is considered under Requirement R4.5-A(1).

RR-0774H: *Provide more predefined exception names with finer granularity.* Rejected: insufficient user benefit (grouping exceptions). See Section 4.5-A(1). This issue was given thorough consideration in the original design, and insufficient evidence is given in this RR to justify reconsidering the decision.

RR-0774I: *Create separate standards, such as X-Windows, SQL.* Rejected: The creation of separate standards is outside the scope of the Ada 9X revision effort. See Section 13.1.

RR-0774J: *Allow generic parameters for any Ada entity, e.g., exceptions.* This request is considered under Study Topic S4.4-A(1).

RR-0774K: *Allow subprograms as parameters.* This request is considered under Requirement R4.1-B(1).

RR-0774L: *Allow pragma INTERFACE within a package body.* Rejected: The compiler needs this information before a package body is compiled in order to minimize the need for recompilation. See Section 13.1.

RR-0774M: *Allow a subprogram to be renamed in a body.* Rejected: Since a renaming declaration already is allowed in a body, the intent behind this request is unclear. See Section 13.1.

RR-0774N: *Allow task cleanup on termination of parent.* This request is considered under Study Topic S4.2-A(2). Finalization is one of the matters to be studied.

Appendix B: Numerical Listing of AIs

Comments on Ada 83 that were submitted in response to the Postscript in the Standard and that suggested changes to the language were given Ada Commentary numbers and classified as ''study'' commentaries. These are listed below with an indication of how they affected the requirements. Study commentaries are underlined when they contain examples or discussion that may be especially helpful to the Mapping/Revision Team or when reviewing proposed changes to the language.

AI-00003: *Allow data of mode IN in SEND_ CONTROL.* Rejected: There is no requirement to fix the low-level I/O programming capabilities in the language. Other needs are more important. See Section 13.1.

AI-00140: *Allow -1..10 as a discrete range in loops.* This request is considered under Section A.3.12.

AI-00142: *Allow pragma SHARED to be applied to components of composite objects.* This request is considered under Requirement R7.1-A(1).

AI-00211: *Additional control statement to hop to end of the loop.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

AI-00214: *Allow accept statements in program units nested in tasks.* This request makes a useful suggestion for improvement in the ability to modularize code in task bodies. See Section 2.2.7 of this document. This AI provides another example in the spirit of RR-0543.

AI-00216: *Provide standard methods for testing whether characters are numeric, upper case, lower case, control, etc., independent of character representation.* This request is considered under Study Topic S10.2-A(1). This AI requests that such tests be specifiable in a uniform manner, regardless of the representation for a character set.

AI-00223: *Require adequate resolution for the function CLOCK.* This request is considered under Requirement R5.1-A(1).

AI-00262: *Real literals with fixed point multiplication and division.* This request makes a useful suggestion for improvement in the ability to use real literals in fixed point expressions. See Section 2.2.8 of this document. This AI makes the same suggestion as RR-0591.

AI-00274: *Proposed extension of the USE clause — record component visibility.* Rejected: Introducing a Pascal-like **use** clause for records might be convenient, but it is not necessarily straightforward to ensure that all components of the record maintain their existence throughout the scope of the **use** clause. There are more important requirements to be addressed. See Section 13.1.

AI-00280: *Allow pragma OPTIMIZE in package specifications.* This request is considered under Requirement R2.2-C(1).

AI-00285: *Need to be able to access a base numeric type in some algorithms.* This request is considered under Study Topic S4.4-A(1). This AI gives an example of the difficulties of getting access to a numeric base type, which was needed when trying to write TEXT_IO.FIXED_IO in Ada.

AI-00291: *Can't define a generic package that works for all floating point types.* This request is considered under Study Topic S4.4-A(1).

AI-00327: *Instantiating with an incomplete private type.* This request makes a useful suggestion for improvement in the ability to use private types before their full declaration. See Section 2.2.5 of this document. This AI is similar to RR-0542, but contains more detail.

AI-00329: *Look-ahead operation for TEXT_IO.* This request is considered under Requirement R4.6-B(1).

AI-00345: *Record type with variant having no discriminants.* This request is considered under Requirement R4.6-B(1). The principle requested use for untagged variants is for I/O.

AI-00349: *Delete copy-in/copy-back for scalar and access parameters.* This AI is the same as RR-0693.

AI-00378: *Enumeration literals should be made directly visible by a subtype declaration.* This request is considered under Section A.2.3.

AI-00382: *Allow generic subprogram bodies.* This request is considered under Requirement R2.2-B(1).

AI-00390: *Character literals should be made directly visible by a subtype declaration.* This request is considered under Section A.2.3.

AI-00404: *Use of incomplete private types in generic formal part.* This request is considered under Requirement R2.2-C(1). This AI illustrates an unintended annoying consequence of the rule restricting the use of an incomplete private type.

AI-00420: *Allow 256 values for type CHARACTER.* This request is considered under Requirement R3.1-A(1).

AI-00421: *Eliminate pragma ELABORATE.* This request is considered under Section A.2.1. This AI explains the dangers in the current definition of pragma ELABORATE.

B: Numerical Listing of AIs

AI-00427: *Semi-constrained subtypes.* Rejected: not a language issue. See Section 13.7. This comment reflects a misunderstanding of the language.

AI-00429: *Allow array type definition for record component.* Rejected: insufficient user benefit (anonymous arrays as record components). See Section 13.4.4. LSN-222 discusses the potential complexity of allowing this capability. See Language Study Notes, 1983, available from the Ada Information Clearinghouse.

AI-00442: *Time zone information in package CALENDAR.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4. It is not clear that time zone information is of sufficient general use to warrant a change to the language.

AI-00450: *Should allow raising of an exception in another task.* This request is considered under Requirement R5.3-A(1). A different solution is called for.

AI-00451: *Task entries as formal parameters to generics.* This request is considered under Study Topic S4.4-A(1). This AI discusses some workarounds that are needed if entries are not distinguished as generic formal parameters.

AI-00452: *Allow record types as generic formal parameters.* This request is considered under Study Topic S4.4-A(1).

AI-00453: *STORAGE_SIZE for tasks.* This request is considered under Section A.3.5.

AI-00458: *Problem with naming of subunits.* This request is considered under Study Topic S4.3-C(1).

AI-00460: *Allow non-integral powers for exponentiation.* Rejected: This change is difficult to make because of the overload resolution rules. This problem was considered extensively in the initial design, and all solutions posed difficulties to users. There are more important changes to focus on in this revision of Ada. See Section 13.1.

AI-00473: *Any form of actual parameter should be allowed as a default parameter.* This request makes a useful suggestion for improvement in the treatment of subprogram parameters. See Section 2.2.3 of this document. This AI points out an easily fixed inconsistency in the language.

AI-00477: *Case choices should not have to be static.* Rejected: too much implementor change for the payoff (non-static case labels). See Section 13.5.2.

AI-00478: *Allow reading of OUT formal parameters.* This request is considered under Section A.3.10. This AI points out that if a formal **out** parameter is used as an actual **out** parameter in a call, it is quite natural to want to read the returned value before returning from a call.

AI-00479: *Initialize access type OUT parameters to null.* This request is considered under Section A.3.10. This AI is essentially the same as RR-0559.

AI-00480: *Operators should be made directly visible by a subtype declaration.* This request is considered under Section A.2.3.

AI-00485: *Having independent standard input and output files is not useful for interactive I/O.* This request is considered under Requirement R4.6-A(1).

AI-00487: *END_OF_PAGE and END_OF_FILE should not return TRUE when there is still an empty line to be read.* This request is considered under Requirement R4.6-B(1).

AI-00488: *Skipping of leading line terminators in GET routines causes problems in interactive I/O.* This request is considered under Requirement R4.6-A(1).

AI-00510: *Use ISO symbols and standards in the Ada ISO Standard.* This request is considered under User Need U3.1-A. This commentary also requests that national alphabets be usable in identifiers, character literals, string literals, and comments. All of these requests are addressed by Requirements R3.1-A(1-5).

AI-00518: *Fixed and floating type declarations needlessly different.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

AI-00519: *Default SMALL should be a power of two times the range.* This request is considered under Requirement R2.2-C(1). This request reflects the need for fixed point types with maximum accuracy for the specified range. This need is in conflict with the Information Systems need for maximum range with only the specified accuracy.

AI-00521: *Fixed point subtypes should not inherit SMALL.* Rejected: insufficient information in the RR to evaluate properly. See Section 13.3. The submitted comment does not give enough motivation for the suggested change to understand why a change might be useful.

AI-00526: *Rounding up or down.* This request is considered under Requirement R2.4-A(1).

AI-00529: *Resolving the meaning of an attribute name.* Rejected: The rules for resolving the overloading of an attribute prefix were adopted after considerable review of complex cases. The example given in this AI does not suggest that there is sufficient user need to reconsider this complicated area of the language. See Section 13.1.

AI-00538: *Declaring constant arrays with an anonymous type.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

AI-00539: *Allow use of array/record attributes in representation clauses.* This request makes a useful suggestion for improvement in the ability to use representation attributes. See Section 2.2.4 of this document.

AI-00540: *Completing a private type declaration with a subtype declaration.* This request is considered under Section A.4.2.

AI-00544: *File ''append'' capability proposed* . This AI is the same as RR-0405.

AI-00545: *Procedure to find if a file exists.* This AI is the same as RR-0404.

AI-00570: *Releasing heap storage associated with task type instances.* This request is considered under Section A.1.1. Implementations would find it easier to return unused storage if tasks could not exist outside their masters.

AI-00572: *Unique path name for subunits.* This AI is the same as RR-0402.

AI-00582: *Need a standard name for null address.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

AI-00584: *Restrict argument of RANGE attribute in Ada 9x.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4. The example motivating the change is only one of several ways to write implementation-dependent programs and does not justify the proposed language change.

AI-00595: *Name of the current exception.* This AI is the same as RR-0403.

AI-00600: *Why we need unsigned integers in Ada.* This request is considered under Requirement R6.1-A(1).

AI-00605: *GET_LINE skips terminators at the end of the line, which is inconsistent with other GET procedures.* This request is considered under Requirement R4.6-B(1).

AI-00609: *Floating point machine attributes inadequate to fully characterize machine characteristics.* This request is considered under Study Topic S11.1-B(1).

AI-00681: *Can't declare a constant of a NULL record type.* Rejected: insufficient user benefit to justify disturbing the language. See Section 13.4.

AI-00812: *Attributes SAFE_LARGE and SAFE_ SMALL should be static.* This request is considered under Requirement R2.2-C(1).

AI-00832: *Effect of depending on parameter passing method when calling non-Ada programs.* This request is considered under Requirement R2.3-A(2).

AI-00840: *Allow access OUT parameter as attribute prefix.* This request makes a useful suggestion for improvement in the treatment of subprogram parameters. See Section 2.2.3 of this document. This AI points out an unneeded and overly restrictive rule.

AI-00850: *Rejecting a unit when a pragma's assumptions are not met.* This request is considered under Study Topic S2.3-A(1). This AI is being actively considered by the Ada Rapporteur Group.

AI-00851: *Need easy direct visibility to the equality operations.* This AI is the same as RR-0694.

AI-00852: *Exiting blocks.* This AI is the same as RR-0695.

AI-00853: *Pragma INLINE should not apply to all overloads.* This AI is the same as RR-0687.

AI-00854: *Related packages need access to a private type's representation.* This AI is the same as RR-0684.

AI-00855: *Limit recompilation effect when compiling a subprogram body as a library unit.* This AI is the same as RR-0688.

AI-00856: *Optional bodies should not be unlinked without a warning.* This AI is the same as RR-0689.

AI-00857: *Priorities of interrupts.* This AI is the same as RR-0686.

AI-00858: *Functions implemented in machine code.* This AI is the same as RR-0691.

AI-00859: *Pragmas LIST and PAGE should be optional.* This AI is the same as RR-0696.

AI-00873: *Type conversion/qualification of undefined scalar values.* This request is considered under Requirement R2.3-A(2). The AI suggests a simple upward-compatible rule change that is consistent with the intent of the Requirement.

AI-00874: *Ensure that access values are values of 'ADDRESS.* This request is considered under Requirement R6.4-A(1).

Appendix C: Included Documents

C.1 Why Static Expressions Can't Contain Explicit Conversions

The Ada 83 restriction that forbids explicit conversions in static expressions stems from a comment submitted during the development of Ada. This comment is reproduced below since these comments are not easily accessible to most people.

!section *04.09 (10) Gary Dismukes/TeleSoft 82-10-11* *#4709.*
!version *16*
!class *Amendment: Resolution*
!topic *May rounding be nondeterministic?*

For conversions from real values to integer values it is stated that ''if the operand is halfway between two integers ..... rounding may be either up or down.'' This suggests that the rounding operation may vary from evaluation to evaluation, even for the same operand. Given that this is true, and given that such conversions may appear within static expressions it appears that the following examples may (or may not) be legal:

Example 1:

```
type SMALL_1 is range 1 .. 1;

S : SMALL_1;

case S is
    when SMALL_1( 1.5 ) => null;
end case;
```

Example 2:

```
type SMALL_2 is range 1 .. 2;

S : SMALL_2;

case S is
  when SMALL_2( 1.5 ) => null;
  when SMALL_2( 1.5 ) => null;
end case;
```

In the first example, the case choice may be legal or illegal depending on how the compiler evaluates the static conversion. In the second example the statement may or may not be legal depending on whether the compiler evaluates the conversions as being equal or as being 1 and 2. In this case, even if the statement is accepted it is nondeterministic as to which case alternative is executed. Note that the legality of these examples may be nondeterministic even for recompilation on the same implementation.

RESPONSE*: Yes, rounding may be nondeterministic.*

The legality of a case statement can depend on the implementation in other ways (e.g. by using FLOAT'DIGITS).

The problems of nondeterminism is serious enough to justify the removal of conversions from static expressions.

C.2 Restriction on Negative Literals

The restriction forbidding the use of negative literals in certain contexts such as **for** I **in** -1 .. 10 **loop** stems from the following comment submitted during the Ada 83 design.

!section *03.06.01 (02) D Taffs 82-08-30* *#3705.*
!version *16*
!class *Amendment: Resolution*
!topic *Implicit conversions of universal discrete ranges*

The wording of the first sentence of this paragraph is unclear, and I think it does not mean what it says. I suggest replacing this sentence with:

> For a discrete range used in a constrained array definition and defined by a range, if BOTH bounds of the range are (potentially) values of the type universal integer, then the predefined type INTEGER is used for the type of each bound.

The current wording says an EXPRESSION defining a bound is implicitly converted. It is not at all clear what happens when an EXPRESSION is implicitly converted, as 4.6 only defines implicit conversion for certain primaries. Note that implicit conversion is not listed among the references for 3.6.1, although "conversion" and "subtype conversion" are. I believe that the intent of 3.6.1(2) is that the BOUNDS of the RANGE be predefined INTEGER, rather than applying an implicit conversion to predefined INTEGER and using whatever type results. For example, in the following example, I think the index range should be of type INTEGER, using the first "+" (and implicitly converting 2 to MY_INT). The range should not be MY_INT, using the third "+" (if only the 2's are implicitly converted) or INTEGER, using universal_integer "+" (if the expression is implicitly converted, as the current wording suggests):

```
type MY_INT is range ... ;

function "+" (R : MY_INT) return INTEGER;
function "+" (X : INTEGER) return INTEGER;   -- hides INTEGER "+"
function "+" (R : INTEGER) return MY_INT;

...

Q : array ("+"(R=>2) .. "+"(R=>2)) of ...
```

If my interpretation is correct, then in some cases (as above) no implicit conversion to INTEGER need be applied at all (the above example converts both 2's to MY_INT, in order to define the range as INTEGER). Therefore, the wording must be changed (as above) to provide a particular context, rather than apply a particular implicit conversion. The current wording makes no sense at all for the above example.

Also, it is not clear whether that sentence applies if EITHER bound is universal integer, or if BOTH bounds are universal integer. This should be made clearer, by saying "if BOTH bounds of the range" in the sentence.

RESPONSE*: The intent of this feature is to cover cases such as*

```
array(1 .. 10);
```

We should keep it simple. For this reason we will restrict the feature to

- *integer literals*
- *named numbers*

- *attributes*

(see version 19)

COMMENT *03.06.01 (02) D Taffs 83-01-02 #5435.*
!version *21*
!class *Amendment: No Action*
!topic *array (-1..1) is now illegal (see #3705)*

.. since -1 is not a numeric literal. The drastic solution taken here is unnecessary, since the response to #3705 assumed that two overloading resolution passes would be required, but this is not the case. Allowing the 1 in -1 to be implicitly converted to INTEGER in the above example is no more complex than allowing -1 to be converted for

```
array (-1..A+B)
```

where A+B resolves to type INTEGER. The suggested wording in 3705 was only intended to say that if both bounds are of type universal integer before implicit conversions are applied, then each bound is to be considered to occur in a context requiring an INTEGER expression. The downward pass in overloading resolution will then select the correct implicit conversions. Taking this approach seems preferable to ruling out bounds that are negative integer literals (requiring named numbers in this case).

RESPONSE: Arrays with negative literal bounds are very rare, and, in our opinion, not worth complicating the rules.

Implicit conversions and overload resolutions are interweaved, so that the phrase "are of type universal_integer before implicit conversions" has little meaning in the general case: subexpressions of a universal expression may require implicit conversions (such as

```
BOOLEAN'POS(F = 1) or, simpler 2**3).
```

Appendix D: KWIC Listing of RR and AI Titles

This Appendix contains a KWIC (KeyWord In Context) Index to the titles of the Revision Requests and the AIs. An individual entry is structured as follows:

RR-xxxx Sec Tail Before Keyword Head

The RR or AI number appears in the first column followed by the section number (of the Rationale) in which the RR is listed. The *Head* and *Tail* sections may be blank. If not, the title is read in the order ‘‘*Head*, *Before*, *Keyword*, *Tail*’’. If part of the title is too long, a slash (/) indicates where the title has been truncated. The section number says ‘‘met’’ if the RR is satisfied in a general way; see Appendix A for further information on such RRs.

RR-0049	13.4	notation when the same name is on both sides of	:=	Allow special
RR-0541	4.2	Allow user-defined	:=, =, DESTROY operations to support memory mngmnt	
RR-0682	13.5.1	Allow user-defined overloaded operators such as	‘‘?’’, ‘‘:-’’, etc	
RR-0534	2.2.13	Allow brackets other than	‘‘(’’, ‘‘)’’ in aggregates, etc	
RR-0663	4.2	Allow certain overloading of	:= and subscripting	
RR-0765	13.1	Allow ‘‘when Package_Name.others’’	=> as exception handler	
RR-0392	13.5	Need ‘‘semi-limited’’ type with predefined	:= but no predefined =	
RR-0682	13.5.1	Allow user-defined overloaded operators such as ‘‘?’’,	‘‘:-’’, etc	
RR-0413	4.2	Allow user-written	:= for all types	
RR-0755	2.2.13	Allow ‘‘[’’ instead of	‘‘(’’ for indexed components	
RR-0534	2.2.13	Allow brackets other than ‘‘(’’,	‘‘)’’ in aggregates, etc	
RR-0755	2.2.13	Allow	‘‘[’’ instead of ‘‘(’’ for indexed components	
RR-0609	4.2	Allow user-defined override of =, /=,	:= on all types	
RR-0669	4.2	Allow user-written	:= routines	
RR-0011	13.6	Expression	0**0 should not be 1 as this is an indeterminate form	
RR-0700	13.1	Ensure that constant functions like sin(	10.0) are evaluated at compile-time	
AI-00140	12.3.12	Allow -	1..10 as a discrete range in loops	
RR-0739	2.2	Relax	11.6 canonical order rules to allow more optimization	
RR-0683	2.2	replacements are allowed Section	11.6 of the Standard is unclear about what	
RR-0387	2.2	more optimizing Relax	11.6 optimization rules to allow compiler to do	
RR-0685	2.2	Clarify and loosen	11.6 to allow more optimization	
RR-0130	4.6	Replace DEFAULT_xy variables in Chapter	14 by functions	
RR-0294	met	for embedded applications; make Chapter	14 optional	I/O packages are not suitable
RR-0134	13.6	Require re-evaluation of entry’count on	abandoned entries	
RR-0335	5.3	Effect of	abort statement is too implementation-dependent	
RR-0063	5.3	Protect tasks from being	aborted while performing critical functions	
RR-0770	13.7	Make	aborting yourself cause instant completeness	
RR-0166	13.3	Allow definition of the literal representations of an	abstract data type	
RR-0167	4.3	Allow classes of	abstract data types	
RR-0217	13.1	Require that a parameter of an entry be used within an	accept	
RR-0771	9.3	Require tasks to have an	accept for each entry	
RR-0216	9.3	Require that each task entry have at least one	accept statement	
RR-0658	5.2	entry call Allow	accept statement possibility in a conditional	
RR-0499	12.3.2	Like other ‘‘blocks’’, allow exception handlers in	accept statements	
RR-0498	5.2	Make selective wait symmetrical with respect to	accept statements and entry calls	
AI-00214	2.2.7	Allow	accept statements in program units nested in tasks	
RR-0543	2.2.7	Allow	accept statements in subprograms nested inside tasks	
RR-0580	2.2.7	inside tasks Allow	accepts within subprograms/packages nested	
AI-00285	4.4	Need to be able to	access a base numeric type in some algorithms	
RR-0560	4.3	related packages Need to	access a private type’s representation in	
RR-0261	2.3	Need compile-time warnings for	access before elaboration errors	
RR-0640	6.1	Need to	access chunk of a bit vector as a whole	
RR-0039	11.2	Make it easier to	access FORTRAN libraries	
RR-0098	13.4	Generalize incomplete typing for types other than	access or private	
AI-00840	2.2.3	Allow	access OUT parameter as attribute prefix	
RR-0684	4.3	Related packages need	access to a private type’s representation	
RR-0104	12.1.1	Prohibit	access to a task outside its master	
RR-0774E	4.5	Provide	access to context of an exception situation	
RR-0110	6.4	Provide explicit control over placement of and	access to data in different types or regions of/	
RR-0286B	5.2	run-time system Embedded system user may need	access to interrupts that are also used by the	
RR-0559	13.6	If allow reading of OUT parameters, initialize OUT	access to NULL	
AI-00479	12.3.10	Initialize	access type OUT parameters to null	
RR-0197	13.6	designated object cannot be modified For	access types, parameter mode IN should mean the	

RR-0287 2.4 Make access types point directly to designated object
RR-0338 6.4 and safe conversion between ADDRESS values and access values /pointers to static objects
AI-00874 6.4 Ensure that access values are values of ’ADDRESS
RR-0258 6.4 Need access values that point to declared objects
RR-0238 6.4 Allow access values to designate read-only memory
RR-0293 6.4 Allow access values to point to declared objects
RR-0247 13.6 Don’t initialize access variables by default to NULL
RR-0276 5.1 Need user specified accuracy and precision control over timing
RR-0225 11.1 Ensure floating point representation with desired accuracy is used
RR-0401 2.2 cannot be done efficiently because of accuracy requirements /fixed-point operations
RR-0352 5.1 Require Calendar.Clock to return consistently accurate local system time
RR-0068 2.4 The Standard should explicitly acknowledge that I/O support is optional for embedded/
RR-0647 4.1 Need ability to select actions depending on state without using case statements
RR-0013 2.1 task execution Allow task activation to occur at a higher priority than
RR-0497 13.7 /default discriminants for types used as generic actual can yield a surprising run-time error
RR-0720 11.1 Floating-point model should reflect actual hardware architectures
RR-0252E 11.1 Provide a floating point model that reflects actual machine architecture
RR-0169 13.4 Allow ‘‘null’’ procedures for actual or default generic formal subprogram values
AI-00473 2.2.3 Any form of actual parameter should be allowed as a default parameter
RR-0239B 2.2.3 A renamed type cannot be used in an actual parameter type conversion
RR-0749 12.3.11 /allow index sliding for slices serving as actual parameters and as values in record/
RR-0586 4.4 /of the same generic unit may have to evaluate their actual parameters in different orders
RR-0549 4.4 Ensure the use of unconstrained actual types is always legal
AI-00582 13.4 Need a standard name for null address
AI-00874 6.4 Ensure that access values are values of ’ ADDRESS
RR-0302 2.4 should define literals for values of type ADDRESS The language
RR-0388 4.1 Proposal for clean way of executing a subprogram by its address
RR-0291 6.4 Clarify whether use of an address clause causes storage to be initialized
RR-0114 6.3 just on the type Allow an address clause for each task instance, and not
RR-0374 4.2 distributed systems Ada should address memory management requirements in
RR-0086 13.4 Need to initialize a record component to the address of the record itself
RR-0195 6.3 Need interrupt address per task, not task type
RR-0421B 6.3 memory address structure; a single/ Interrupt address structure is sometimes different from
RR-0338 6.4 /pointers to static objects and safe conversion between ADDRESS values and access values
RR-0349 6.3 different and should not be treated/ Interrupt addresses and memory addresses are conceptually
AI-00223 5.1 Require adequate resolution for the function CLOCK
RR-0105 5.1 Allow application to set/ adjust clocks
RR-0198 13.4 Allow positional aggregate for single-component aggregate
RR-0571B 2.1 Clarify the effect when the choice in an aggregate is outside the range of the/
RR-0341 2.2 Allow discriminant value in record aggregate to be non-static
RR-0391 13.4 Clumsy syntax for based numbers, especially in aggregates
RR-0573 12.3.11 initialization and as components of record aggregates /aggregates for record component
RR-0240 12.3.11 Non-sliding aggregates and slices in component associations
RR-0605 12.2.5 Rules for OTHERS in aggregates are confusing
RR-0534 2.2.13 Allow brackets other than ‘‘(’’, ‘‘)’’ in aggregates, etc
RR-0053 13.4 Allow aggregates for null records and arrays
RR-0573 12.3.11 and as components of/ Slide indices of array aggregates for record component initialization
AI-00329 4.6 Look- ahead operation for TEXT_IO
RR-0774F 2.2 Allow aliased exceptions within the same exception handler
RR-0463 13.4 ’Size is unclear; perhaps need ’Spacing and ’ Allocation
RR-0417 6.2 Length clause should force allocation of EXACT number of bits
RR-0665B 8.2 Support allocation of parallel processes to processors
RR-0176 9.1 Document run-time system performance and memory allocation strategies
RR-0370E 4.2 control blocks when tasks are created by an allocator Need to recover space for task
RR-0107 5.1 to specify clock timing interval if hardware allows this flexibility Allow application
RR-0339 13.1 Support sorting in extended alphabets
RR-0549 4.4 Ensure the use of unconstrained actual types is always legal
RR-0626 6.2 /by SEQUENTIAL_IO and DIRECT_IO are not portable among compilers, even for the same target/
RR-0521 5.2 convenient support for use of shared memory among tasks Need more
RR-0041 4.3 Allow overloaded subunits with respect to a common ancestor library unit
RR-0769 2.1 Correct wording in the definition of ancestor unit
RR-0321 13.4 record components Permit anonymous array and record declarations for
RR-0617 13.6 Eliminate anonymous array types
RR-0443 13.4.4 Need for anonymous array types as record components
RR-0672 13.4 Need anonymous pointer types
AI-00538 13.4 Declaring constant arrays with an anonymous type
RR-0345 13.1 Need standardized interface to other ANSI languages
RR-0420 4.6 Need file ‘‘extend’’ or ‘‘append’’ capability

RR-0207	4.6	Add TEXT_IO support with Exists function and	Append procedure
RR-0405	4.6	Need convenient way to	append to a file
RR-0382	4.6	Need to be able to rename and	append to a file in standard Ada
AI-00142	7.1	Allow pragma SHARED to be	applied to components of composite objects
RR-0581B	2.1	no body Clarify the effect of	applying pragma ELABORATE to a package that has
RR-0252E	11.1	point model that reflects actual machine	architecture Provide a floating
AI-00584	13.4	Restrict	argument of RANGE attribute in Ada 9x
RR-0355	2.4	Standardize means of getting the OS command line	arguments
RR-0635	13.4	Provide basic support for extended precision integer	arithmetic
RR-0144	13.3	not present Require support for fixed point	arithmetic even if floating point hardware is
RR-0634	6.1	Provide	arithmetic shift operations for integers
RR-0731	11.1	floating point/ Use the Language Compatible	Arithmetic Standard as a basis for Ada's
RR-0557	4.3	/to provide subprogram bodies helps get	around the inability to overload subunit names
RR-0334	7.2	its work domain, e.g., to process part of an	array /to specify task parameters giving a task
RR-0573	12.3.11	initialization and as/ Slide indices of	array aggregates for record component
RR-0321	13.4	Permit anonymous	array and record declarations for record components
RR-0336	13.4.4	type definitions in records; nice for array-of-	array case Allow array
RR-0122	2.2	implementation to reject some integer types as	array indexes Permit an
RR-0713	4.4	Relax	array matching rules for generics
RR-0668	4.3	Need package types to get, for example, an	array of packages
RR-0017	6.2	Be able to treat an Ada object as an	array of storage units
RR-0308	11.1	Add libraries for	array processing
RR-0251	13.6	Invent new notations to distinguish function call,	array reference, and conversions
RR-0133	7.2	Allow a task component of an	array to get its index
RR-0513	12.3.9	of = for any type, e.g., returning an	array type Allow overloading
AI-00429	13.4.4	Allow	array type definition for record component
RR-0336	13.4.4	array-of-array case Allow	array type definitions in records; nice for
RR-0617	13.6	Eliminate anonymous	array types
RR-0443	13.4.4	Need for anonymous	array types as record components
RR-0418	2.2	Representation clauses for	array types need to be added
RR-0336	13.4.4	Allow array type definitions in records; nice for	array-of-array case
AI-00539	2.2.4	Allow use of	array/record attributes in representation clauses
RR-0053	13.4	Allow aggregates for null records and	arrays
RR-0139	6.1	Provide shift and rotate operations for boolean	arrays
RR-0323	13.4.2	Generalize slice for multidimensional	arrays
RR-0494	13.4.2	Allow slices for any dimension in multidimensional	arrays
RR-0508	13.4.2	Allow slices for any dimension in multidimensional	arrays
RR-0520	13.1	Language should distinguish ''sequence'' and ''mapping''	arrays
RR-0257	2.1	Ensure that BOOLEAN and BYTE	arrays can be tightly packed
RR-0426C	13.6	Omitting index constraint in constant	arrays causes programmer errors
RR-0726	6.4	Need non-contiguous	arrays, static pointers
RR-0510	2.2.10	Re-indexing	arrays via type conversions
AI-00538	13.4	Declaring constant	arrays with an anonymous type
RR-0018	6.4	Need pre-elaborated constant	arrays with variable-sized elements
RR-0619	2.2	Eliminate three replacement characters, stick to normal	ASCII
RR-0736	3.1	Need 8-bit	ASCII in Ada
RR-0148	3.1	for extended and graphic characters (256	ASCII set) Provide support
RR-0043	13.1.1	Make it easier and more portable to use	assembler with Ada
RR-0088	4.2	Problems associated with user-defined	assignment
RR-0551	4.6	Need	assignment capability for TEXT_IO.FILE_TYPE
RR-0001	4.2	Limited types need	assignment, constants
RR-0070	4.2	Allow user-defined	assignment for limited types
RR-0160	4.2	Allow user-defined	assignment for limited types
RR-0201B	4.2	Overload the	assignment operation
RR-0202	4.2	mode rules for limited types that have an	assignment operation Relax parameter
RR-0163	10.4	strings with appropriate equality and	assignment operations /for variable-length
RR-0184	4.2	Need user-defined	assignment operator for limited private type
RR-0212	13.6	Allow	assignment to record discriminant like other components
RR-0310	10.4	Need convenient way to pad with blanks in string	assignments
RR-0421C	6.3	objects, not task types Need to	associate interrupts with entries of task
RR-0240	12.3.11	Non-sliding aggregates and slices in component	associations
RR-0571A	12.2.5	Allow use of OTHERS choice with named	associations when index bounds are determined by/
RR-0029	12.2.5	determined by/ Allow use of OTHERS with named	associations when the index constraint is
AI-00850	2.3	Rejecting a unit when a pragma's	assumptions are not met
RR-0710	6.3	operating system Need to tie task entries to	asynchronous external events generated by
RR-0183	5.4		Asynchronous inter-task communication is not available
RR-0655	5.4	Add	asynchronous message queues
RR-0748	5.4	Provide standard package of	asynchronous primitives

RR-0106 5.3 Provide asynchronous transfer of control
RR-0083 5.3 call/selective wait construct Provide asynchronous transfer of control via entry
RR-0768 5.3 Need to asynchronously interrupt another task to stop it
RR-0742 5.3 Need ability to asynchronously stop another task
RR-0434 7.1 Need atomic read/write operations on shared volatile memory
RR-0623 12.3.3 Define RANGE attribute for discrete ranges
RR-0495 13.6 Remove leading space in the result of the 'IMAGE attribute for integers
RR-0454 11.1 Need Entier function or attribute for real types
RR-0059 2.2.14 underlying value Need an attribute for returning a representation's
RR-0155 12.3.3 Define RANGE attribute for scalar types
RR-0304 12.3.3 Define RANGE attribute for scalar types
AI-00584 13.4 Restrict argument of RANGE attribute in Ada 9x
AI-00529 13.1 Resolving the meaning of an attribute name
AI-00840 2.2.3 Allow access OUT parameter as attribute prefix
RR-0298 2.1 Clarify classes of objects usable as attribute prefixes
RR-0453 11.1 Provide a special function or attribute yielding the sign of a numeric value
RR-0613 13.4.1 problems with implementation-defined attributes /attributes solve portability
RR-0674 13.4.1 Allow user-defined attributes as functions
RR-0664 12.3.1 Need 'IMAGE and 'VALUE attributes for floating-point types
RR-0715 2.2 Allow user-defined type conversions and attributes for numeric types
RR-0716 11.1 Unify and add attributes for numeric types
RR-0509 13.4.1 Allow user-defined attributes for user-defined or private types
RR-0406 13.4.1 Allow user-defined attributes for user-defined types
AI-00539 2.2.4 Allow use of array/record attributes in representation clauses
AI-00609 11.1 machine characteristics Floating point machine attributes inadequate to fully characterize
RR-0048 2.2.4 Extend static expressions to include representation attributes of composite types
AI-00812 2.2 Make attributes SAFE_LARGE and SAFE_SMALL static
RR-0613 13.4.1 implementation-defined attributes User-defined attributes solve portability problems with
RR-0317 2.2.12 Augment Ada's looping: over reals, list items, etc
RR-0553 4.6 GET_LINE should not automatically call SKIP_LINE
RR-0475 4.2 reclaim storage Need automatically-invoked user-defined routines to
RR-0286C 5.2 Run-time system should avoid entering privileged mode
RR-0368A 4.3 Ensure unnecessary recompilation is avoided
RR-0638 13.1 specified explicitly Axioms for built-in operations should be
RR-0636 11.1 Improve Ada's axioms for floating point operations
RR-0401 2.2 efficiently because of accuracy/ Mixed- base fixed-point operations cannot be done
AI-00285 4.4 Need to be able to access a base numeric type in some algorithms
RR-0190 4.4 Allow use of a base type within a generic unit
RR-0511 4.4 Allow use of a base type within a generic unit
RR-0127 13.4 Allow real number output in non-decimal bases
RR-0569 12.2.2 Relax rules separating basic from later declarative items
RR-0594 12.2.2 Relax rules separating basic from later declarative items
RR-0635 13.4 Provide basic support for extended precision integer arithmetic
RR-0731 11.1 Use the Language Compatible Arithmetic Standard as a basis for Ada's floating point model
RR-0205 12.3.13 Allow program unit name on PRIVATE, BEGIN, and EXCEPTION
RR-0735 6.3 Need ability to change interrupt bindings at run-time
RR-0310 10.4 Need convenient way to pad with blanks in string assignments
RR-0773 6.2 Need to pack variable-length records into a block for data transmission
RR-0695 13.1.2 Allow EXIT from block for legibility
RR-0491 13.1.2 Code would be clearer if one could EXIT from a block statement
RR-0632 13.1.2 Allow EXIT from a block statement for consistency
RR-0711 4.6 I/O by a task in multi-task application should not block whole program
RR-0499 12.3.2 Like other ''blocks'', allow exception handlers in accept statements
RR-0370E 4.2 Need to recover space for task control blocks when tasks are created by an allocator
AI-00382 2.2 Allow generic subprogram bodies
RR-0562 4.4 compilation of generic specifications and bodies Require separate
RR-0557 4.3 /use of renaming declarations to provide subprogram bodies helps get around the inability to/
RR-0689 12.2.4 Optional bodies should not be unlinked without a warning
RR-0764 12.4.1 Allow subprogram bodies to be defined by RENAMES
RR-0550 12.4.1 Allow subprogram bodies to be defined by RENAMES or generic instantiation
RR-0093 13.4 of deferred constants to be given in a package body Allow full declaration
RR-0157 12.4.1 Allow renaming when defining a subprogram body
RR-0214 13.1 Require that a subprogram parameter be used within the body
RR-0231 12.4.1 Allow a rename definition of a subprogram body
RR-0307 4.3 of private declarations to be in the package body Allow completion
RR-0470 12.4.1 or generic instantiation to define a subprogram body Allow renaming
RR-0581B 2.1 pragma ELABORATE to a package that has no body Clarify the effect of applying
RR-0688 4.3 required when redeclaring a subprogram body Unnecessary recompilation

RR-0774L 13.1 Allow pragma INTERFACE within a package body
RR-0774M 13.1 Allow a subprogram to be renamed in a body
RR-0547 2.2 subprograms, allow merge of specification/ body for generic subprograms Like non-generic
RR-0604 2.2 subprograms, allow merge of specification/ body for generic subprograms Like non-generic
RR-0725 12.4.1 Need rename in package body for routine in package specification
RR-0426A 12.2.4 The effect of an optional package body is confusing to users
RR-0268 13.6 Separation of specification and body is not worth it
RR-0262 13.7 Do not require existence of subunit for body stubs
RR-0426B 2.2 Allow declaration and body to be combined for generic subprograms
RR-0364 12.4.1 Allow a subprogram body to be defined by generic instantiation
RR-0055 12.4.1 Allow a subprogram body to be defined by renaming or generic instantiation
RR-0666 12.4.1 Allow a subprogram body to be given by generic instantiation
RR-0667 12.4.1 Allow a subprogram body to be given by RENAMES
RR-0096B 12.4.1 Allow a procedure body to be provided by a renaming declaration
RR-0257 2.1 Ensure that BOOLEAN and BYTE arrays can be tightly packed
RR-0139 6.1 Provide shift and rotate operations for boolean arrays
RR-0571A 12.2.5 choice with named associations when index bounds are determined by context /use of OTHERS
RR-0252D 2.2 Fixed point type should include the bounds of the range definition
RR-0191 2.2 Fixed point model numbers should include the bounds of the type definition
RR-0566 2.2 Fixed point model numbers should include the bounds of the type definition
RR-0644 9.1 Standard should specify time bounds/constraints for certain operations
RR-0534 2.2.13 Allow brackets other than ‘‘(’’, ‘‘)’’ in aggregates, etc
RR-0263 13.1 CONSTRAINT_ERROR is too broadly defined
RR-0450 6.4 Need efficient manipulation of buffers whose type is determined at run time
RR-0729 9.1 way to turn off optimization to eliminate bugs Language should provide
RR-0540 4.3 Allow a new package to build on an existing package
RR-0638 13.1 Axioms for built-in operations should be specified explicitly
RR-0257 2.1 Ensure that BOOLEAN and BYTE arrays can be tightly packed
RR-0050 3.1 Provide multi-national and multi- byte characters
RR-0766 6.1 Allow bit-wise operations (AND, SHIFT) on integers, bytes, etc
RR-0252B 11.1 whether rounding or truncation is used in real calculations Programmer needs to know/control
AI-00442 13.4 Time zone information in package CALENDAR
RR-0280 5.1 must be/ Short delays are too inefficient; Calendar time unnecessary; timing performance
RR-0352 5.1 local system time Require Calendar.Clock to return consistently accurate
RR-0458 4.4 way to escape into weakly typed subprogram call Need convenient
RR-0658 5.2 statement possibility in a conditional entry call Allow accept
RR-0697 5.2 Allow entry call alternative in selective wait
RR-0251 13.6 Invent new notations to distinguish function call, array reference, and conversions
RR-0659 4.4 Need to make entry call on a generic formal parameter
RR-0060 2.2.9 Allow inlining of subprograms from some but not all call sites
RR-0553 4.6 GET_LINE should not automatically call SKIP_LINE
RR-0014 4.1 Need to call subprograms loaded in ROM
RR-0064 4.1 Allow some form of subprogram callback
AI-00832 2.3 Effect of depending on parameter passing method when calling non-Ada programs
RR-0158 13.3 Allow multi-way conditional and timed entry calls
RR-0498 5.2 with respect to accept statements and entry calls Make selective wait symmetrical
RR-0629 4.1 and function types for use in subprogram calls Need procedure
RR-0708 13.5 Allow infix function calls
RR-0471 13.6 Allow specification of parameter modes in subprogram calls for clarity
RR-0076 5.2 based on priorities Allow selection of entry calls from entry queues and open alternatives
RR-0421D 6.3 /of interrupts as ordinary, timed, or conditional calls may depend inappropriately on the/
RR-0083 5.3 Provide asynchronous transfer of control via entry call/selective wait construct
RR-0336 13.4.4 definitions in records; nice for array-of-array case Allow array type
RR-0199 12.3.13 Allow IF, CASE, and SELECT constructs to be named
AI-00477 13.5.2 Case choices should not have to be static
AI-00216 10.2 /characters are numeric, upper case, lower case, control, etc., independent of character/
RR-0340 12.3.13 Allow optional simple name on CASE, IF, and SELECT statements
AI-00216 10.2 /testing whether characters are numeric, upper case, lower case, control, etc., independent of/
RR-0359 4.6 Allow mixed case output for enumeration literals
RR-0650 13.5.2 Allow non-static case statement choices, non-discrete case statement expression
RR-0320 13.5 Generalize case statement for other types, including REAL
RR-0312 13.5 Generalize case statement to decision table
RR-0561 13.5.2 Allow case statement to operate on strings for string processing
RR-0647 4.1 select actions depending on state without using case statements Need ability to
RR-0621C 13.4 Allow case statements to dispatch on value of an exception
RR-0135 13.4 for intermediate results Catenation should not raise CONSTRAINT_ERROR
RR-0535 11.1 Provide CEILING and FLOOR numeric operators
RR-0358 11.1 Need support for floor, ceiling, truncate, and whole operations

RR-0733 13.5 Need fixed-point types not centered on zero
RR-0150 13.7 memory requirements Provide ‘‘chaining’’ of different programs to reduce
RR-0020 5.2 program execution, so priorities should be changeable /of functions may change during
RR-0130 4.6 Replace DEFAULT_xy variables in Chapter 14 by functions
RR-0294 met not suitable for embedded applications; make Chapter 14 optional I/O packages are
AI-00420 3.1 Allow 256 values for type CHARACTER
RR-0331 3.1 Need predefined LONG_ CHARACTER (16 bits) and LONG_LONG_CHAR (32)
RR-0390 3.1 Need 8-bit unsigned CHARACTER for Greek and graphics symbols
RR-0096A 12.2.3 Permit renaming an enumeration literal as a character literal
AI-00390 12.2.3 visible by a subtype declaration Character literals should be made directly
RR-0528 2.1 Change Ada character names to recognized names for verbal comm.
AI-00216 10.2 case, lower case, control, etc., independent of character representation /are numeric, upper
RR-0201A 13.5.1 Liberalize overloading of operators to other character sequences
RR-0034 3.1 Ada should use ISO 8859/1-9 (8-bit) character set
RR-0438 3.1 Allow use of multi-octet character set
RR-0311 3.1 Generalize character set for 8-bit characters
RR-0367 3.1 Need support for national language character sets, including string comparison
AI-00609 11.1 inadequate to fully characterize machine characteristics /point machine attributes
RR-0252A 11.1 point standard; allow full use of machine characteristics /support for IEEE floating
AI-00609 11.1 Floating point machine attributes inadequate to fully characterize machine characteristics
RR-0050 3.1 Provide multi-national and multi-byte characters
RR-0311 3.1 Generalize character set for 8-bit characters
RR-0148 3.1 Provide support for extended and graphic characters (256 ASCII set)
AI-00216 10.2 Provide standard methods for testing whether characters are numeric, upper case, lower case,/
RR-0330 3.1 Allow national characters in literals, comments, and identifiers
RR-0619 2.2 Eliminate three replacement characters, stick to normal ASCII
RR-0574 2.2 Inability to eliminate constraint check for OUT parameters
RR-0301 2.1 can be improved The wording concerning checking for consistency between compilations
RR-0584 4.4 instantiation is given Need stricter checking of formal generic subtypes when an
RR-0554 9.1 I/O input Need constraint checks for target of Unchecked_Conversion and
RR-0571B 2.1 the applicable/ Clarify the effect when the choice in an aggregate is outside the range of
RR-0571A 12.2.5 bounds are determined by/ Allow use of OTHERS choice with named associations when index
RR-0650 13.5.2 Allow non-static case statement choices, non-discrete case statement expression
AI-00477 13.5.2 Case choices should not have to be static
RR-0252C 11.1 Ensure programmer can choose appropriate floating point representation
RR-0103B 6.2 means of reading large data structures in chunks Provide efficient
RR-0265 13.7 Allow implementations to short- circuit in general, forget AND THEN
RR-0662 4.3 Need package classes and inheritance for object-oriented programming
RR-0271 13.6 CONTROLLED or STATIC Distinguish storage classes for variables with key words like
RR-0167 4.3 Allow classes of abstract data types
RR-0298 2.1 Clarify classes of objects usable as attribute prefixes
RR-0730 13.4 part of a package should have its own context clause The private
RR-0291 6.4 Clarify whether use of an address clause causes storage to be initialized
RR-0699 13.3 Do not treat an unaccepted length clause for a type as an error
RR-0114 6.3 the type Allow an address clause for each task instance, and not just on
RR-0288 13.5 Integrate representation clause information with declarations
RR-0581C 13.4 to mention a package name given in the context clause of a parent library unit /for a subunit
RR-0200 12.3.4 Allow optional when_ clause on RAISE and RETURN statements
RR-0362 12.3.4 Allow optional when_ clause on the raise statement
RR-0417 6.2 of bits Length clause should force allocation of EXACT number
RR-0588 13.4 Provide a form of USE clause that hides outer homographs
RR-0555 12.2.3 a type Need ‘‘selective’’ USE clause to get just operators and subprograms of
AI-00274 13.1 Proposed extension of the USE clause — record component visibility
AI-00539 2.2.4 Allow use of array/record attributes in representation clauses
RR-0095 13.4 Allow applicable units to be named in USE clauses and pragma ELABORATE
RR-0065 4.3 To improve reuse possibilities, allow rep clauses and various pragmas to be separated from/
RR-0418 2.2 Representation clauses for array types need to be added
RR-0411 2.4 Express record representation clauses in a machine-independent way
RR-0565 2.2 ’SMALL is unsuitably defined; need for representation clauses inappropriate
RR-0290 13.5 The syntax used in record representation clauses is hard to read
RR-0677 2.2.2 Allow initialization clauses on scalar type declarations
RR-0171 4.3 Allow target-dependent code (including rep clauses) to be separate from other code
RR-0274 2.1 The visibility rules could be explained more clearly
AI-00223 5.1 Require adequate resolution for the function CLOCK
RR-0037 5.2 using simulated time rather than a real-time clock Allow tasks (i.e., delays) to execute
RR-0107 5.1 Allow application to specify clock timing interval if hardware allows this flexibility
RR-0105 5.1 Allow application to set/adjust clocks
RR-0439 4.2 Require automatic garbage collection

RR-0643 4.2 Garbage collection can now be done well; encourage its use
RR-0019 4.2 procedures for safely controlling use of collections /types to specify finalization
RR-0507 11.2 Provide information/control over row-major or column-major ordering
RR-0426B 2.2 Allow declaration and body to be combined for generic subprograms
RR-0355 2.4 Standardize means of getting the OS command line arguments
RR-0709 2.4 Need more portability in getting command line inputs
RR-0330 3.1 Allow national characters in literals, comments, and identifiers
RR-0746 13.7 Allow pictures/graphics as comments in source code
RR-0181 8.1 Need standard means of communicating between Ada programs
RR-0222 8.1 predefined packages for process control/ communication Need additional
RR-0528 2.1 character names to recognized names for verbal communication Change Ada
RR-0665C 5.4 Support message-driven intertask communication
RR-0587 5.4 Provide for communication between loosely coupled tasks
RR-0378 8.1 Need standard means of communication in distributed system
RR-0183 5.4 Asynchronous inter-task communication is not available
RR-0224 8.1 Add communication support required for distributed systems
RR-0367 3.1 language character sets, including string comparison Need support for national
RR-0731 11.1 Ada's floating point model Use the Language Compatible Arithmetic Standard as a basis for
RR-0343 13.4 Provide better facilities for conditional compilation
RR-0356 13.4 Need a way to get the compilation date and time within a program
RR-0692 2.3 pragmas are not/ /pragmas to cause unsuccessful compilation if restrictions implied by the
RR-0237 4.3 Make separate compilation independent of a particular library model
RR-0562 4.4 Require separate compilation of generic specifications and bodies
RR-0283 4.3 Need convenient way to set global compilation parameters
RR-0091 4.3 Don't specify the compilation process in the Standard
RR-0757 2.1 Clean up definitions of program unit and compilation unit
RR-0154 13.1 Subunits should not have to be at the outermost compilation unit level
RR-0545 13.1 Subunits should not have to be at the outermost compilation unit level
RR-0065 4.3 /clauses and various pragmas to be separated from the compilation unit to which they apply
RR-0607 13.1 Allow names of compilation units to be overloadable, operator symbols
RR-0242 2.3 Require compilation warnings for potential run-time errors
RR-0301 2.1 The wording concerning checking for consistency between compilations can be improved
RR-0279 2.2 If tasks are not used, the run-time system and compiled code should not include code for/
RR-0177 4.3 Standardize interface between compiler and library for configuration management
RR-0175 5.2 Define interface between compiler- and target-specific run-time system aspects
RR-0386 9.1 Need standard way of telling the compiler not to optimize
RR-0387 2.2 Relax 11.6 optimization rules to allow compiler to do more optimizing
RR-0209 2.3 Require the compiler to report certain-to-be-raised exceptions
RR-0368B 4.3 by tools other than those provided by the compiler vendor /the library can be manipulated
RR-0353 2.4 Unchecked conversion should eliminate compiler-dependent fields
RR-0003 4.2 Provide a compiler-independent finalization mechanism
RR-0062 7.1 Ensure memory mapped devices are treated correctly by compilers
RR-0626 6.2 /and DIRECT_IO are not portable among compilers, even for the same target machine e.g.,/
RR-0616 2.3 constraint errors Require compilers to diagnose statically-detectable
RR-0328 9.3 Require compilers to report questionable uses of the language
RR-0211 2.3 Require compilers to report unrecognized or incorrect pragmas
RR-0700 13.1 functions like sin(10.0) are evaluated at compile-time Ensure that constant
RR-0165 2.3 Allow parameter constraint violations to be compile-time errors
RR-0639 8.2 Need compile-time initialization of complex data structures
RR-0261 2.3 elaboration errors Need compile-time warnings for access before
RR-0244B 2.3 Flag run-time errors at compile-time when possible
AI-00540 12.4.2 subtype declaration Completing a private type declaration with a
RR-0208 13.4 and SEQ_IO operations without waiting for completion /to initiate TEXT_IO, DIRECT_IO,
RR-0305 2.1 Clarify wording of FOR loop completion
RR-0542 2.2.5 another allow usage of private type before its completion declaration One way or
RR-0307 4.3 package body Allow completion of private declarations to be in the
AI-00429 13.4.4 Allow array type definition for record component
RR-0198 13.4 Allow positional aggregate for single- component aggregate
RR-0240 12.3.11 Non-sliding aggregates and slices in component associations
RR-0462 12.3.7 even when the selected/ Allow selected component form of type mark in a formal part
RR-0462 12.3.7 /type mark in a formal part even when the selected component has the same identifier as the/
RR-0707 2.2.6 Need same-name component identifiers in different variants
RR-0573 12.3.11 Slide indices of array aggregates for record component initialization and as components of/
RR-0133 7.2 Allow a task component of an array to get its index
RR-0577 2.2 /deferred constant of composite type having a component of an incompletely declared private/
RR-0679 13.4 Allow component selection on objects of a private type
RR-0086 13.4 Need to initialize a record component to the address of the record itself
AI-00274 13.1 Proposed extension of the USE clause — record component visibility

RR-0212 13.6 Allow assignment to record discriminant like other components
RR-0321 13.4 array and record declarations for record components Permit anonymous
RR-0381 2.2 Records should have composed operations with respect to components
RR-0443 13.4.4 Need for anonymous array types as record components
RR-0749 12.3.11 as actual parameters and as values in record components /index sliding for slices serving
RR-0755 2.2.13 Allow ‘‘[’’ instead of ‘‘(’’ for indexed components
RR-0532 2.2.6 Allow same-type record components in different variants to share name
AI-00142 7.1 Allow pragma SHARED to be applied to components of composite objects
RR-0524 6.4 Allow functions to return references to components of objects; allow programmer to ensure/
RR-0573 12.3.11 /aggregates for record component initialization and as components of record aggregates
RR-0381 2.2 Records should have composed operations with respect to components
AI-00142 7.1 Allow pragma SHARED to be applied to components of composite objects
RR-0119 7.1 Need synchronized reference to elements of shared composite objects
RR-0577 2.2 Allow deferred constant of composite type having a component of an incompletely/
RR-0048 2.2.4 to include representation attributes of composite types Extend static expressions
RR-0718 9.1 Need predictable results in numeric computation, especially regarding optimization
RR-0349 6.3 Interrupt addresses and memory addresses are conceptually different and should not be/
RR-0174 4.3 Allow packages to be generic with respect to concurrency protection
RR-0132 12.3.4 with EXIT statement Allow optional WHEN <condition> on RAISE statement for consistency
RR-0141 12.3.4 Allow WHEN <condition> on RAISE statements
RR-0614 12.3.4 value clearer Allow WHEN condition RETURN to make selection of returned
RR-0158 13.3 Allow multi-way conditional and timed entry calls
RR-0421D 6.3 /treatment of interrupts as ordinary, timed, or conditional calls may depend inappropriately on/
RR-0343 13.4 Provide better facilities for conditional compilation
RR-0658 5.2 Allow accept statement possibility in a conditional entry call
RR-0518 13.1 Provide syntax to declare subprogram pre/post conditions
RR-0177 4.3 Standardize interface between compiler and library for configuration management
RR-0344 2.2 Need to simplify/relax the conformance rules
RR-0631 2.2 Make conformance rules consistent
RR-0426C 13.6 Omitting index constraint in constant arrays causes programmer errors
AI-00538 13.4 Declaring constant arrays with an anonymous type
RR-0018 6.4 Need pre-elaborated constant arrays with variable-sized elements
RR-0329 13.1 Using a deferred constant before it has a value
RR-0246 8.2 time when initialized with static/ Ensure that constant declarations are not elaborated at run
RR-0452 13.4 overloadable constants) Allow constant functions in static expressions (or
RR-0700 13.1 at compile-time Ensure that constant functions like sin(10.0) are evaluated
AI-00681 13.4 Can’t declare a constant of a NULL record type
RR-0577 2.2 of an incompletely declared/ Allow deferred constant of composite type having a component
RR-0001 4.2 Limited types need assignment, constants
RR-0452 13.4 in static expressions (or overloadable constants) Allow constant functions
RR-0313 13.4 Allow deferred constants of arbitrary (i.e., non-private) types
RR-0611 4.1 Allow subprogram types, variables, constants, parameters, etc
RR-0451 4.3 Changes to package constants should not cause recompilation
RR-0093 13.4 Allow full declaration of deferred constants to be given in a package body
RR-0100 13.4 Allow constants to use default values to get value
RR-0653 8.2 Need to declare constants whose value is supplied after linking
RR-0006 4.4 Distinguish unconstrained/ constrained generic formal types
RR-0472 4.4 Distinguish unconstrained/ constrained generic formal types
AI-00427 13.7 Semi- constrained subtypes
RR-0473 13.5 Allow ‘‘partially’’ constrained subtypes of discriminated records
RR-0446 4.4 Tighten the contract model by distinguishing constrained/unconstrained generic types
RR-0571B 2.1 is outside the range of the applicable index constraint /when the choice in an aggregate
RR-0574 2.2 Inability to eliminate constraint check for OUT parameters
RR-0554 9.1 Unchecked_Conversion and I/O input Need constraint checks for target of
RR-0616 2.3 Require compilers to diagnose statically-detectable constraint errors
RR-0426C 13.6 Omitting index constraint in constant arrays causes programmer errors
RR-0029 12.2.5 /use of OTHERS with named associations when the index constraint is determined by context
RR-0165 2.3 Allow parameter constraint violations to be compile-time errors
RR-0399 13.1 Break up overly broad predefined exceptions, e.g., CONSTRAINT_ERROR
RR-0583 2.1 Delete NUMERIC_ERROR if now subsumed under CONSTRAINT_ERROR
RR-0135 13.4 Catenation should not raise CONSTRAINT_ERROR for intermediate results
RR-0263 13.1 CONSTRAINT_ERROR is too broadly defined
RR-0644 9.1 Standard should specify time bounds/ constraints for certain operations
RR-0567 2.2 Allow variable declaration to get constraints from initial value
RR-0660 4.2 Need constructors and destructors for package types
RR-0292 2.1 Section 13.6 of the standard has no semantic content
RR-0029 12.2.5 when the index constraint is determined by context /use of OTHERS with named associations
RR-0571A 12.2.5 when index bounds are determined by context /OTHERS choice with named associations

RR-0730 13.4 The private part of a package should have its own context clause
RR-0581C 13.4 subunit to mention a package name given in the context clause of a parent library unit /for a
RR-0282 13.3 Ada program structure hides important context information
RR-0774E 4.5 Provide access to context of an exception situation
RR-0726 6.4 Need non- contiguous arrays, static pointers
RR-0031 13.5.3 Provide a way to test for a value in a non- contiguous set
RR-0446 4.4 constrained/unconstrained generic/ Tighten the contract model by distinguishing
AI-00003 13.1 Allow data of mode IN in SEND_ CONTROL
RR-0106 5.3 Provide asynchronous transfer of control
RR-0015 5.2 Allow task priorities to control all queuing/select decisions
RR-0347 5.2 /applications to change priorities under program control; allow task priority to increase as a/
RR-0370E 4.2 Need to recover space for task control blocks when tasks are created by an allocator
RR-0759 13.3 Add real-time and verification facilities for control engineering
AI-00216 10.2 /characters are numeric, upper case, lower case, control, etc., independent of character/
RR-0774C 4.3 Extend control of library unit visibility
RR-0575 2.2.9 Need better (more selective) control over inlining
RR-0110 6.4 different types or regions of/ Provide explicit control over placement of and access to data in
RR-0507 11.2 Provide information/ control over row-major or column-major ordering
RR-0121 5.2 Provide more user control over scheduling decisions
RR-0737 5.2 select statement Allow reliable user control over selection of alternatives in a
RR-0276 5.1 Need user specified accuracy and precision control over timing
AI-00211 13.4 Additional control statement to hop to end of the loop
RR-0615 2.2.12 Define LOOP/UNTIL control structure as in Pascal
RR-0286A 5.2 Embedded system users need the ability to control timer utilities
RR-0083 5.3 Provide asynchronous transfer of control via entry call/selective wait construct
RR-0457 4.3 Structure library units as groups, control visibility of library units
RR-0252B 11.1 in real calculations Programmer needs to know/ control whether rounding or truncation is used
RR-0222 8.1 Need additional predefined packages for process control/communication
RR-0361 4.6 Increase the number of options for controlling the output format of numbers
RR-0019 4.2 to specify finalization procedures for safely controlling use of collections Allow types
RR-0239B 2.2.3 type cannot be used in an actual parameter type conversion A renamed
RR-0724 2.1 resolution rules, especially for implicit conversion Need clearer/simpler overload
RR-0734 12.3.11 Generalize cases that allow implicit subtype conversion
RR-0554 9.1 Need constraint checks for target of Unchecked_ Conversion and I/O input
RR-0338 6.4 Provide pointers to static objects and safe conversion between ADDRESS values and access/
RR-0103A 2.2.3 Allow unchecked conversion for IN OUT and OUT parameters
RR-0476 13.6 target type Allow user-written type- conversion functions with the same name as the
RR-0449 13.1 Do not allow unchecked conversion of private types
RR-0353 2.4 Unchecked conversion should eliminate compiler-dependent fields
RR-0009 12.3.6 discrete expression Allow static conversion to static discrete type of static
AI-00873 2.3 Type conversion/qualification of undefined scalar values
RR-0251 13.6 distinguish function call, array reference, and conversions Invent new notations to
RR-0510 2.2.10 Re-indexing arrays via type conversions
RR-0715 2.2 Allow user-defined type conversions and attributes for numeric types
RR-0099 12.3.6 Explicit type conversions should be allowed in static expressions
RR-0062 7.1 Ensure memory mapped devices are treated correctly by compilers
RR-0134 13.6 Require re-evaluation of entry' count on abandoned entries
RR-0275 2.2 Error-prone and counter-intuitive aspects of RENAMES
RR-0544 4.2 Need indivisible update on reference counts
RR-0370E 4.2 space for task control blocks when tasks are created by an allocator Need to recover
RR-0435 9.3 standard for simple Ada subset for safety- critical applications Need secondary
RR-0063 5.3 Protect tasks from being aborted while performing critical functions
RR-0206 2.1 Paragraph numbers should be included in the cross references
RR-0309 2.1 Ensure all cross references are complete and correct
RR-0389 13.4 There is a need for ''cyclic'' discrete types in the language
RR-0431 5.3 A terminate alternative cannot be used to stop cyclic tasks
RR-0270 13.4 Allow specification of read-only data from a package
RR-0110 6.4 /control over placement of and access to data in different types or regions of memory
AI-00003 13.1 Allow data of mode IN in SEND_CONTROL
RR-0678 7.1 Pragma SHARED is not sufficient for data shared between programs; need VOLATILE
RR-0639 8.2 Need compile-time initialization of complex data structures
RR-0103B 6.2 Provide efficient means of reading large data structures in chunks
RR-0773 6.2 pack variable-length records into a block for data transmission Need to
RR-0166 13.3 of the literal representations of an abstract data type Allow definition
RR-0167 4.3 Allow classes of abstract data types
RR-0356 13.4 Need a way to get the compilation date and time within a program
RR-0656 5.1 Need timed exceptions for deadline scheduling
RR-0127 13.4 Allow real number output in non- decimal bases

RR-0357	10.1	Need packed decimal, wide-ranging fixed-point,	decimal deltas
AI-00378	12.2.3	should be made directly visible by a subtype	declaration Enumeration literals
AI-00390	12.2.3	should be made directly visible by a subtype	declaration Character literals
AI-00480	12.2.3	Operators should be made directly visible by a subtype	declaration
AI-00540	12.4.2	Completing a private type declaration with a subtype	declaration
RR-0096B	12.4.1	Allow a procedure body to be provided by a renaming	declaration
RR-0096C	12.4.2	of a private type to be provided by a renaming	declaration Allow the full declaration
RR-0542	2.2.5	usage of private type before its completion	declaration One way or another allow
RR-0690	12.4.2	and private types to be completed by subtype	declaration Allow incomplete
RR-0426B	2.2	Allow	declaration and body to be combined for generic sbprgrms
RR-0096C	12.4.2	a renaming declaration Allow the full	declaration of a private type to be provided by
RR-0010	2.2	discriminants to be a derived/ Allow the full	declaration of a private type with
RR-0093	13.4	in a package body Allow full	declaration of deferred constants to be given
RR-0082	2.2.5	visible package specification Allow	declaration of objects of private types in
RR-0094	2.2	Make the multiple	declaration rules more complete and consistent
RR-0567	2.2	Allow variable	declaration to get constraints from initial value
AI-00540	12.4.2	Completing a private type	declaration with a subtype declaration
RR-0267	2.1	confusing in distinguishing specifications and	declarations The Standard is
RR-0288	13.5	Integrate representation clause information with	declarations
RR-0448	4.3	sets of subprograms to depend on common	declarations Allow different
RR-0677	2.2.2	Allow initialization clauses on scalar type	declarations
RR-0727	12.2.3	Need selective direct visibility of package	declarations
RR-0032	12.2.2	Allow grouping of variable	declarations and related subprograms
RR-0259	13.7	Incomplete type	declarations are dangerous and unnecessary
RR-0246	8.2	when initialized with/ Ensure that constant	declarations are not elaborated at run time
RR-0429	12.2.3	Need construct that makes just overloadable	declarations directly visible
RR-0321	13.4	Permit anonymous array and record	declarations for record components
RR-0005	4.4	sharing unnecessarily difficult Exception	declarations in generic packages make code
RR-0428	12.2.2	Order of	declarations is too restrictive
RR-0753	13.6	Make syntax for task type	declarations more consistent
AI-00518	13.4	Fixed and floating type	declarations needlessly different
RR-0423	2.2	Remove discriminant restriction on full	declarations of private types
RR-0425	13.1	Need open ranges in	declarations of real subtypes
RR-0601	2.2	Allow library-level	declarations to be defined by RENAMES
RR-0307	4.3	Allow completion of private	declarations to be in the package body
RR-0557	4.3	get around the inability/ The use of renaming	declarations to provide subprogram bodies helps
RR-0569	12.2.2	Relax rules separating basic from later	declarative items
RR-0594	12.2.2	Relax rules separating basic from later	declarative items
AI-00681	13.4	Can't	declare a constant of a NULL record type
RR-0653	8.2	Need to	declare constants whose value is supplied after linking
RR-0712	4.4	generic unit Need ability to	declare double precision numeric types within a
RR-0517	9.3	Provide syntax to	declare program units free from side-effects
RR-0518	13.1	Provide syntax to	declare subprogram pre/post conditions
RR-0022	12.2.3	Need direct visibility of operators	declared in another package
RR-0370A	8.2	Can't recover space	declared in library units when reconfiguring a system
RR-0258	6.4	Need access values that point to	declared objects
RR-0293	6.4	Allow access values to point to	declared objects
RR-0577	2.2	type having a component of an incompletely	declared private type /constant of composite
RR-0427	12.1.1	Do not permit a function to return a locally-	declared task object
RR-0652	12.2.3	operator directly visible	Declaring a subtype should make the equality
AI-00538	13.4		Declaring constant arrays with an anonymous type
RR-0269	13.6	Make subprograms not recursive by	default
RR-0497	13.7	actual can yield a surprising/ Presence of	default discriminants for types used as generic
RR-0576	13.4	Allow parameter	default expressions to make use of previous IN parameters
RR-0447	4.6	Need to be able to preserve/restore the	default file at any point
RR-0169	13.4	Allow ''null'' procedures for actual or	default generic formal subprogram values
RR-0350	2.1	Clarify wording dealing with	default initial values
RR-0595	2.2.2	Allow	default initialization for all types
RR-0649	2.2.2	Allow	default initialization for all types (not just records)
RR-0161	2.2.2	Allow	default initialization for any non-limited type
RR-0129	2.2.2	non-limited type Allow	default initialization to be specified for any
RR-0714	12.3.8	Allow	default names for all generic formal parameters
AI-00473	2.2.3	Any form of actual parameter should be allowed as a	default parameter
RR-0097	13.4	Allow/require explicit action to get	default parameter value
RR-0007	2.4	should be specified	Default representation for enumeration types
AI-00519	2.2	the range	Default SMALL should be a power of two times
RR-0247	13.6	Don't initialize access variables by	default to NULL
RR-0100	13.4	Allow constants to use	default values to get value

RR-0484 4.6 generic TEXT_IO packages Add DEFAULT_xy functionality as parameters to
RR-0130 4.6 Replace DEFAULT_xy variables in Chapter 14 by functions
RR-0120 4.2 space is exhausted Allow users to defer the signalling of STORAGE_ERROR when
RR-0329 13.1 Using a deferred constant before it has a value
RR-0577 2.2 component of an incompletely declared/ Allow deferred constant of composite type having a
RR-0313 13.4 non-private) types Allow deferred constants of arbitrary (i.e.,
RR-0093 13.4 Allow full declaration of deferred constants to be given in a package body
RR-0116 5.2 priorities needed for mode change and graceful degradation User-modifiable
RR-0192 5.2 priorities during mode change and for graceful degradation Need ability to change
RR-0384 5.1 which causes an exception after specified delay Cannot write subprogram
RR-0612 13.3 Should allow both delay and terminate alternatives in selective wait
RR-0421A 6.3 Need to delay in processing an interrupt
RR-0281 2.1 Confusing treatment of term ''delay statement''
RR-0280 5.1 unnecessary; timing performance must be/ Short delays are too inefficient; Calendar time
RR-0037 5.2 than a real-time clock Allow tasks (i.e., delays) to execute using simulated time rather
RR-0253 13.1 DIGITS and DELTA approach leads to inefficiency, non-portability
RR-0256 13.1 Fixed-point approach with range and delta is not what is needed
RR-0357 10.1 Need packed decimal, wide-ranging fixed-point, decimal deltas
RR-0421D 6.3 /as ordinary, timed, or conditional calls may depend inappropriately on the run-time system
RR-0448 4.3 Allow different sets of subprograms to depend on common declarations
RR-0459 2.4 for interoperability; lessen implementation dependence Improve support
RR-0042 2.3 Clarify the meaning of incorrect-order dependence and its effects
RR-0066 2.3 with erroneous execution/incorrect order dependences Reduce risks associated
RR-0143 9.1 Document implementation dependences
RR-0179 6.3 The treatment of interrupts is too implementation- dependent
RR-0335 5.3 Effect of abort statement is too implementation- dependent
RR-0236 2.4 documented whenever/ Reduce implementation- dependent behavior, or at least, ensure it is
RR-0171 4.3 separate from other code Allow target- dependent code (including rep clauses) to be
RR-0353 2.4 Unchecked conversion should eliminate compiler- dependent fields
RR-0582 4.5 /for getting additional implementation- dependent info about state when an exception is raised
RR-0215 2.1 Clarify termination of tasks dependent on library packages
RR-0124 5.2 Ensure that code dependent on task scheduling algorithms is portable
AI-00832 2.3 calling non-Ada programs Effect of depending on parameter passing method when
RR-0647 4.1 Need ability to select actions depending on state without using case statements
RR-0558 13.4 Deriver of type should be able to hide subset of derived operations
RR-0010 2.2 of a private type with discriminants to be a derived type Allow the full declaration
RR-0080 13.3 Derived types are clumsy
RR-0482 4.3 needed operations Multiple derived types from same package do not generate
RR-0052 4.3 desired operations Multiple derived types from same package do not give
RR-0599 4.3 Certain changes to derived/private types will help inheritance
RR-0558 13.4 of derived operations Deriver of type should be able to hide subset
RR-0238 6.4 Allow access values to designate read-only memory
RR-0287 2.4 Make access types point directly to designated object
RR-0197 13.6 For access types, parameter mode IN should mean the designated object cannot be modified
RR-0225 11.1 Ensure floating point representation with desired accuracy is used
RR-0052 4.3 Multiple derived types from same package do not give desired operations
RR-0541 4.2 Allow user-defined :=, =, DESTROY operations to support memory management
RR-0660 4.2 Need constructors and destructors for package types
RR-0450 6.4 Need efficient manipulation of buffers whose type is determined at run time
RR-0029 12.2.5 named associations when the index constraint is determined by context Allow use of OTHERS with
RR-0571A 12.2.5 with named associations when index bounds are determined by context /use of OTHERS choice
RR-0485 4.6 to get the line length of an input or output device Provide means
RR-0062 7.1 Ensure memory mapped devices are treated correctly by compilers
RR-0077 13.4 Provide stream I/O for digital signal processing
RR-0253 13.1 inefficiency, non-portability DIGITS and DELTA approach leads to
RR-0564 11.1 Allow implementation freedom to include more mantissa digits in floating point safe numbers
RR-0494 13.4.2 Allow slices for any dimension in multidimensional arrays
RR-0508 13.4.2 Allow slices for any dimension in multidimensional arrays
RR-0354 13.6 Introduce dimensional mathematics into the language
RR-0745 13.6 Add facilities for dimensional mathematics to the language
RR-0624 12.2.3 Provide selective direct visibility into a package
RR-0393 12.2.3 operator by renaming Can't get direct visibility of fixed point mult and div
RR-0022 12.2.3 another package Need direct visibility of operators declared in
RR-0232 12.2.3 Need to allow direct visibility of operators in packages
RR-0727 12.2.3 Need selective direct visibility of package declarations
RR-0057 12.2.3 Need direct visibility to infix operators in another package
RR-0474 12.2.3 and operators of a type Need direct visibility to just enumeration literals
RR-0694 12.2.3 Need easy direct visibility to the equality operations

RR-0208 13.4 waiting for/ Need ability to initiate TEXT_IO, DIRECT_IO, and SEQ_IO operations without
RR-0626 6.2 even for/ Files produced by SEQUENTIAL_IO and DIRECT_IO are not portable among compilers,
RR-0593 4.6 Mandate implementation of variant record I/O in DIRECT_IO/SEQUENTIAL_IO
RR-0278 5.2 Tasking model should support common scheduling disciplines more easily
RR-0046 13.5.3 Allow testing in discontiguous ranges and create true sets
RR-0603 13.5.3 Allow discontiguous subtypes of discrete types
RR-0058 13.5.3 Allow discontiguous subtypes of enumeration types
RR-0650 13.5.2 Allow non-static case statement choices, non-discrete case statement expression
RR-0009 12.3.6 conversion to static discrete type of static discrete expression Allow static
RR-0744 2.2.12 Allow for loop to have non-discrete (fixed-point) parameter
AI-00140 12.3.12 Allow -1..10 as a discrete range in loops
RR-0623 12.3.3 Define RANGE attribute for discrete ranges
RR-0522 2.2 Allow non-discrete record discriminants
RR-0009 12.3.6 Allow static conversion to static discrete type of static discrete expression
RR-0363 12.3.1 and ’IMAGE to apply to real types as well as discrete types Allow ’VALUE
RR-0603 13.5.3 Allow discontiguous subtypes of discrete types
RR-0389 13.4 There is a need for ‘‘cyclic’’ discrete types in the language
RR-0289 6.2 views of a record structure even when no discriminant is present Need multiple
RR-0212 13.6 Allow assignment to record discriminant like other components
RR-0423 2.2 of private types Remove discriminant restriction on full declarations
RR-0341 2.2 Allow discriminant value in record aggregate to be non-static
AI-00345 4.6 Record type with variant having no discriminants
RR-0522 2.2 Allow non-discrete record discriminants
RR-0497 13.7 can yield a surprising/ Presence of default discriminants for types used as generic actual
RR-0264 13.3 Discriminants need to stand out more
RR-0248 13.1 Allow users to specify locations for discriminants that are outside record values
RR-0010 2.2 Allow the full declaration of a private type with discriminants to be a derived type
RR-0473 13.5 Allow ‘‘partially’’ constrained subtypes of discriminated records
RR-0621C 13.4 Allow case statements to dispatch on value of an exception
RR-0503 4.1 Provide subprogram types for dispatcher-style programming
RR-0109 8.1 that are helpful when dealing with a single distributed Ada program Provide Ada semantics
RR-0728 8.1 Need simple Ada run-time system for distributed memory MIMD architectures
RR-0071 13.2 Improve support for heterogeneous distributed processing
RR-0378 8.1 Need standard means of communication in distributed system
RR-0224 8.1 Add communication support required for distributed systems
RR-0374 4.2 address memory management requirements in distributed systems Ada should
RR-0515 4.2 update for specific objects, especially in distributed systems /to request indivisible
RR-0376 13.3 Need special treatment of exceptions in distributed/parallel/multi-processor systems
RR-0393 12.2.3 Can’t get direct visibility of fixed point mult and div operator by renaming
RR-0537 13.6 Separate integer divide and floating divide as in Pascal
RR-0591 2.2.8 Allow fixed-point multiply/divide with universal real operands
AI-00262 2.2.8 Real literals with fixed point multiplication and division
RR-0143 9.1 Document implementation dependences
RR-0176 9.1 allocation strategies Document run-time system performance and memory
RR-0481 2.1 Make Ada documentation available in SGML format
RR-0280 5.1 time unnecessary; timing performance must be documented /are too inefficient; Calendar
RR-0236 2.4 /behavior, or at least, ensure it is documented whenever possible
RR-0334 7.2 /to specify task parameters giving a task its work domain, e.g., to process part of an array
RR-0626 6.2 for the same target machine e.g., because of dope vectors /portable among compilers, even
RR-0712 4.4 Need ability to declare double precision numeric types within a generic unit
AI-00526 2.4 Rounding up or down
RR-0213 2.4 to find out if an implementation rounds up or down Need to be able
RR-0665C 5.4 Support message-driven intertask communication
RR-0373 8.2 Need to be able to dynamically alter a program as it is running
RR-0126 13.4 Allow underscore before ‘‘E’’ in exponents
RR-0278 5.2 support common scheduling disciplines more easily Tasking model should
RR-0051C 10.4 Provide packages for string edit functions
RR-0450 6.4 determined at run time Need efficient manipulation of buffers whose type is
RR-0103B 6.2 structures in chunks Provide efficient means of reading large data
RR-0590 5.2 Need clear, efficient, standard support for mutual exclusion
RR-0241 5.2 Need easier and more efficient support for mutual exclusion
RR-0401 2.2 /fixed-point operations cannot be done efficiently because of accuracy requirements
RR-0117 8.2 Provide pre-elaboratable constructs
AI-00421 12.2.1 Eliminate pragma ELABORATE
RR-0095 13.4 units to be named in USE clauses and pragma ELABORATE Allow applicable
RR-0396 12.2.1 ordering rules to reduce need for pragma ELABORATE Add library unit elaboration
RR-0767 12.2.1 problem without requiring the use of pragma ELABORATE Solve the elaboration order
RR-0581 12.2.1 Rules specifying the position of pragma ELABORATE are error-prone and unhelpful

RR-0581C 13.4 name given in the context/ Allow a pragma ELABORATE for a subunit to mention a package
RR-0546 12.2.1 It is too difficult to ensure that pragma ELABORATE is used when it is needed
RR-0581A 12.2.1 Eliminate need for pragma ELABORATE; pragma NOT_ ELABORATE might help
RR-0004 12.2.1 Pragma ELABORATE should be transitive
RR-0233 12.2.1 Pragma ELABORATE should be transitive
RR-0581B 2.1 Clarify the effect of applying pragma ELABORATE to a package that has no body
RR-0246 8.2 Ensure that constant declarations are not elaborated at run time when initialized with/
RR-0018 6.4 Need pre- elaborated constant arrays with variable-sized elements
RR-0245 8.2 Change Standard to encourage pre- elaboration
RR-0285 8.2 Minimize the need for run-time elaboration
RR-0261 2.3 Need compile-time warnings for access before elaboration errors
RR-0244A 8.2 Require pre- elaboration of some constructs
RR-0218 12.2.1 Make the implementation find a good library-unit elaboration order
RR-0767 12.2.1 use of pragma ELABORATE Solve the elaboration order problem without requiring the
RR-0396 12.2.1 pragma ELABORATE Add library unit elaboration ordering rules to reduce need for
RR-0243 8.2 Allow/require elaboration prior to run time
RR-0294 met I/O packages are not suitable for embedded applications; make Chapter 14 optional
RR-0188 6.1 and bit-wise logical operations on integer/ Embedded applications need unsigned integers
RR-0318 2.1 version of the Standard available (with embedded mark-up) Make a machine-readable
RR-0286B 5.2 interrupts that are also used by the run-time/ Embedded system user may need access to
RR-0286A 5.2 control timer utilities Embedded system users need the ability to
RR-0068 2.4 acknowledge that I/O support is optional for embedded systems /Standard should explicitly
RR-0723 8.2 Need support for reconfiguration in emergency cases
AI-00487 4.6 should not return TRUE when there is still an empty line to be read /and END_OF_FILE
AI-00605 4.6 other GET/ GET_LINE skips terminators at the end of the line, which is inconsistent with
AI-00211 13.4 Additional control statement to hop to end of the loop
RR-0596 12.3.13 Allow END type_name to substitute for END RECORD
RR-0673 12.3.13 Allow ‘‘END RECORD type_name’’ to substitute for ‘‘END RECORD’’
RR-0596 12.3.13 Allow END type_name to substitute for END RECORD
AI-00487 4.6 is still an empty line to be/ END_OF_PAGE and END_OF_FILE should not return TRUE when there
RR-0196 5.3 Endorsement of RR-0083
RR-0759 13.3 Add real-time and verification facilities for control engineering
RR-0286C 5.2 Run-time system should avoid entering privileged mode
RR-0454 11.1 Need Entier function or attribute for real types
RR-0186 13.3 It is difficult to write an entire operating system in Ada
RR-0128 4.1 Provide subprograms as parameters to subprograms and entries
RR-0134 13.6 Require re-evaluation of entry’count on abandoned entries
RR-0408 4.4 There is a need for generic formal entries
RR-0628 2.2.11 Need private task entries
AI-00451 4.4 Task entries as formal parameters to generics
RR-0488 4.4 Allow generic formal entries as well as generic formal subprograms
RR-0075 5.2 Queue entries by task priority or FIFO based on application
RR-0487 2.2.11 Need private task entries for exclusive use within the task
RR-0421C 6.3 Need to associate interrupts with entries of task objects, not task types
RR-0710 6.3 generated by operating system Need to tie task entries to asynchronous external events
RR-0090 2.2.11 Allow some task entries to be visible, some not
RR-0771 9.3 Require tasks to have an accept for each entry
RR-0217 13.1 Require that a parameter of an entry be used within an accept
RR-0658 5.2 Allow accept statement possibility in a conditional entry call
RR-0697 5.2 Allow entry call alternative in selective wait
RR-0659 4.4 Need to make entry call on a generic formal parameter
RR-0158 13.3 Allow multi-way conditional and timed entry calls
RR-0498 5.2 with respect to accept statements and entry calls Make selective wait symmetrical
RR-0076 5.2 alternatives based on/ Allow selection of entry calls from entry queues and open
RR-0083 5.3 Provide asynchronous transfer of control via entry call/selective wait construct
RR-0056 met Do not remove task entry families
RR-0216 9.3 Require that each task entry have at least one accept statement
RR-0076 5.2 Allow selection of entry calls from entry queues and open alternatives based on priorities
RR-0657 5.2 Order entry queues based on priority
RR-0134 13.6 Require re-evaluation of entry’count on abandoned entries
RR-0415 5.2 Allow priority inheritance, prioritized entry-queues, and prioritized selective wait
RR-0096A 12.2.3 Permit renaming an enumeration literal as a character literal
RR-0359 4.6 Allow mixed case output for enumeration literals
RR-0474 12.2.3 Need direct visibility to just enumeration literals and operators of a type
RR-0131 13.4 /expression, should have visibility of the enumeration literals of the qualifying type
AI-00378 12.2.3 visible by a subtype declaration Enumeration literals should be made directly
RR-0187 2.4 Need to allow unsigned enumeration representation specifications
RR-0239A 12.2.3 Renaming an enumeration type should make literals visible

RR-0058 13.5.3 Allow discontiguous subtypes of enumeration types
RR-0437 13.5.3 Provide ‘‘supertype’’ capability for merging enumeration types
RR-0007 2.4 Default representation for enumeration types should be specified
RR-0465 2.2.14 Need a way to get the representation from an enumeration value and vice versa
RR-0040 2.2.14 Need a way to determine the internal coding of enumeration values
RR-0220 2.2.14 Need way to get the internal code associated with enumeration values
RR-0732 2.4 Clarify semantics of instantiating ENUMERATION_IO with an integer type
RR-0074 5.2 Define a standard run-time support environment interface
RR-0377 8.2 partitioning of programs for multiple processor environments Ada should allow
RR-0163 10.4 /support for variable-length strings with appropriate equality and assignment operations
RR-0694 12.2.3 Need easy direct visibility to the equality operations
RR-0652 12.2.3 Declaring a subtype should make the equality operator directly visible
RR-0008 12.3.9 Allow overloading of the equality operator for all types
RR-0025 12.3.9 Allow overloading of the equality operator with different operand types
RR-0066 2.3 Reduce risks associated with erroneous execution/incorrect order dependences
RR-0399 13.1 broad predefined exceptions, e.g., CONSTRAINT_ ERROR Break up overly
RR-0497 13.7 generic actual can yield a surprising run-time error /default discriminants for types used as
RR-0699 13.3 an unaccepted length clause for a type as an error Do not treat
RR-0314 13.7 Define minimum-quality error diagnostics in the standard
RR-0135 13.4 Catenation should not raise CONSTRAINT_ ERROR for intermediate results
RR-0583 2.1 Delete NUMERIC_ ERROR if now subsumed under CONSTRAINT_ERROR
RR-0263 13.1 CONSTRAINT_ ERROR is too broadly defined
RR-0118 4.2 Provide a user-specified storage reserve for STORAGE_ ERROR recovery
RR-0120 4.2 Allow users to defer the signalling of STORAGE_ ERROR when space is exhausted
RR-0275 2.2 Error-prone and counter-intuitive aspects of RENAMES
RR-0581 12.2.1 Rules specifying the position of pragma ELABORATE are error-prone and unhelpful
RR-0165 2.3 constraint violations to be compile-time errors Allow parameter
RR-0242 2.3 Require compilation warnings for potential run-time errors
RR-0261 2.3 warnings for access before elaboration errors Need compile-time
RR-0426C 13.6 constraint in constant arrays causes programmer errors Omitting index
RR-0616 2.3 to diagnose statically-detectable constraint errors Require compilers
RR-0244B 2.3 Flag run-time errors at compile-time when possible
RR-0458 4.4 Need convenient way to escape into weakly typed subprogram call
RR-0586 4.4 /of the same generic unit may have to evaluate their actual parameters in different orders
RR-0700 13.1 Ensure that constant functions like sin(10.0) are evaluated at compile-time
RR-0134 13.6 Require re- evaluation of entry’count on abandoned entries
RR-0710 6.3 Need to tie task entries to asynchronous external events generated by operating system
RR-0033A 4.5 Need to find the name of a raised exception
RR-0085 4.5 Need to get the name of the current exception
RR-0205 12.3.13 Allow program unit name on PRIVATE, BEGIN, and EXCEPTION
RR-0219 4.5 raised exception, including an out-of-scope exception /a way to get the name of the last
RR-0400 2.3 allow a task to die silently on an unhandled exception Do not
RR-0403 4.5 Need to be able to get the name of the current exception
RR-0407B 2.3 allow a task to die silently on an unhandled exception Do not
RR-0477 4.5 Provide a way to get the name and location of a raised exception
RR-0526C 4.5 Need to determine the name of a raised exception
RR-0621C 13.4 Allow case statements to dispatch on value of an exception
RR-0384 5.1 Cannot write subprogram which causes an exception after specified delay
RR-0444 13.4 Let the user limit the places where a given exception can be raised
RR-0005 4.4 code sharing unnecessarily difficult Exception declarations in generic packages make
RR-0765 13.1 Allow ‘‘when Package_Name.others =>’’ as exception handler
RR-0774F 2.2 Allow aliased exceptions within the same exception handler
RR-0221 13.4 Need to write common code for group of exception handlers
RR-0499 12.3.2 Like other ‘‘blocks’’, allow exception handlers in accept statements
RR-0621A 4.5 Need to find out which exception has been raised
AI-00450 5.3 Should allow raising of an exception in another task
RR-0651 5.3 Allow one task to raise an exception in another task
RR-0219 4.5 /a way to get the name of the last raised exception, including an out-of-scope exception
RR-0582 4.5 /implementation-dependent info about state when an exception is raised
RR-0145 4.5 Provide a way to get exception name from WHEN OTHERS handlers
RR-0772 4.5 Need to be able to get exception name in a handler
RR-0774G 4.5 Provide exception name in OTHERS handler
RR-0407A 4.5 where raised Need exception name, line number, and unit name
RR-0774H 4.5 Provide more predefined exception names with finer granularity
RR-0774E 4.5 Provide access to context of an exception situation
RR-0036 4.5 to be grouped under a single name by allowing exception subtypes Allow exceptions
RR-0209 2.3 Require the compiler to report certain-to-be-raised exceptions
RR-0228 4.4 Allow generic parameterization with exceptions

RR-0774J 4.4 Allow generic parameters for any Ada entity, e.g., exceptions
RR-0254 9.1 Too much freedom is allowed with respect to exceptions and intermediate expression results
RR-0706 4.4 Allow exceptions and packages as generic parameters
RR-0621B 4.4 Permit exceptions as generic formals
RR-0671 4.4 Allow exceptions as generic parameters
RR-0101B 4.4 Need to pass exceptions as parameters to generic units and subprograms
RR-0526B 4.4 Need to pass exceptions as parameters to generic units and subprograms
RR-0399 13.1 Break up overly broad predefined exceptions, e.g., CONSTRAINT_ERROR
RR-0656 5.1 Need timed exceptions for deadline scheduling
RR-0383 4.4 Need generic exceptions for truly reusable generic units
RR-0376 13.3 Need special treatment of exceptions in/
RR-0490 2.3 Need successful/convenient recovery from exceptions in machine code insertions
RR-0416 4.5 Granularity of predefined exceptions is too coarse
RR-0468 4.4 No generic way to handle exceptions raised by generic formal subprograms
RR-0101A 4.5 Allow exceptions to be grouped under a single name
RR-0526A 4.5 Allow exceptions to be grouped under a single name
RR-0036 4.5 allowing exception subtypes Allow exceptions to be grouped under a single name by
RR-0646 13.2 read in handler Allow exceptions to be parameterized with parameters
RR-0033B 4.4 Need to pass exceptions to subprograms and generic units
RR-0774F 2.2 Allow aliased exceptions within the same exception handler
RR-0504 13.4 Add an exchange operator
RR-0241 5.2 Need easier and more efficient support for mutual exclusion
RR-0590 5.2 Need clear, efficient, standard support for mutual exclusion
RR-0037 5.2 real-time clock Allow tasks (i.e., delays) to execute using simulated time rather than a
RR-0120 4.2 the signalling of STORAGE_ERROR when space is exhausted Allow users to defer
RR-0540 4.3 Allow a new package to build on an existing package
RR-0491 13.1.2 Code would be clearer if one could EXIT from a block statement
RR-0632 13.1.2 Allow EXIT from a block statement for consistency
RR-0695 13.1.2 Allow EXIT from block for legibility
RR-0132 12.3.4 on RAISE statement for consistency with EXIT statement Allow optional WHEN <condition>
RR-0538 9.3 Create new loop structure which bans the EXIT statement
RR-0625 13.6 Change EXIT/WHEN to WHEN/ EXIT to parallel Ada IF and English
RR-0325B 13.6 Allow implementations to experiment with supersets
RR-0274 2.1 The visibility rules could be explained more clearly
RR-0638 13.1 Axioms for built-in operations should be specified explicitly
RR-0068 2.4 optional for embedded/ The Standard should explicitly acknowledge that I/O support is
RR-0024 11.1 decompose floating point numbers into mantissa/ exponent Need a way to
RR-0680 13.1 Predefined exponentiation should take any integer type for exponent
RR-0645 11.1 Need mantissa/ exponent extraction and manipulation
RR-0346 11.1 Need portable way to extract mantissa/ exponent from floating point number
RR-0492 11.1 Decouple mantissa and exponent information in floating point type definitions
AI-00460 13.1 Allow non-integral powers for exponentiation
RR-0680 13.1 Predefined exponentiation should take any integer type for exponent
RR-0126 13.4 Allow underscore before ‘‘E’’ in exponents
RR-0455 4.3 The import and export mechanisms of Ada are too limited
RR-0172 4.3 Make import and export of types easier
RR-0424 13.6 during instantiation Allow names exported from an instance to be redefined
RR-0009 12.3.6 to static discrete type of static discrete expression Allow static conversion
RR-0650 13.5.2 statement choices, non-discrete case statement expression Allow non-static case
RR-0011 13.6 indeterminate form Expression 0**0 should not be 1 as this is an
RR-0254 9.1 with respect to exceptions and intermediate expression results Too much freedom is allowed
RR-0131 13.4 enumeration literals of the/ In a qualified expression, should have visibility of the
RR-0099 12.3.6 Explicit type conversions should be allowed in static expressions
RR-0246 8.2 at run time when initialized with static expressions /declarations are not elaborated
RR-0519 13.1 Simplify overload rules for ambiguous/universal expressions
RR-0705 2.2 performance, remove restrictions on static expressions For better
RR-0452 13.4 Allow constant functions in static expressions (or overloadable constants)
RR-0048 2.2.4 attributes of composite types Extend static expressions to include representation
RR-0576 13.4 Allow parameter default expressions to make use of previous IN parameters
RR-0420 4.6 Need file ‘‘extend’’ or ‘‘append’’ capability
AI-00274 13.1 component visibility Proposed extension of the USE clause — record
RR-0710 6.3 Need to tie task entries to asynchronous external events generated by operating system
RR-0056 met Do not remove task entry families
RR-0111 8.1 Provide explicit support for fault tolerance and recovery
RR-0136 6.1 Provide support for bit- field operations such as shift, rotate
RR-0353 2.4 Unchecked conversion should eliminate compiler-dependent fields
RR-0075 5.2 Queue entries by task priority or FIFO based on application
RR-0405 4.6 Need convenient way to append to a file

RR-0447	4.6	Need to be able to preserve/restore the default	file at any point
RR-0404	4.6	Need convenient way to find out if a particular	file exists
RR-0420	4.6	Need	file ‘‘extend’’ or ‘‘append’’ capability
RR-0382	4.6	Need to be able to rename and append to a	file in standard Ada
AI-00487	4.6	an empty line to be/ END_OF_PAGE and END_OF_	FILE should not return TRUE when there is still
RR-0159	4.6	Add standard package of general	file system functions
RR-0146	13.1	Support for	file/record locking
AI-00485	4.6	Having independent standard input and output	files is not useful for interactive I/O
RR-0626	6.2	are not portable among compilers, even for the/	Files produced by SEQUENTIAL_IO and DIRECT_IO
RR-0552	13.4	Need ‘‘padded’’ line input with truncation and pad-	fill to ’LENGTH
RR-0092	4.2	Allow user-specified	finalization
RR-0385	4.2	Need	finalization code for packages
RR-0203	4.2	Allow	finalization code for packages and tasks
RR-0168	4.2	Allow implicitly-invoked	finalization code for storage management
RR-0466	4.2	release of resources Allow user-defined	finalization for objects of a type to ensure
RR-0523	4.2	release of resources Allow user-defined	finalization for objects of a type to ensure
RR-0003	4.2	Provide a compiler-independent	finalization mechanism
RR-0019	4.2	use of collections Allow types to specify	finalization procedures for safely controlling
RR-0676	4.2	Add	finalization to ensure release of resources
RR-0774H	4.5	Provide more predefined exception names with	finer granularity
RR-0642	13.4	Add label variables to support use of	finite state machines
RR-0249	12.1.2	’	First and ’last for null ranges are defined oddly
RR-0426D	13.4	Optional index in ’	FIRST (and others) causes problems
AI-00518	13.4		Fixed and floating type declarations needlessly different
RR-0144	13.3	hardware is not present Require support for	fixed point arithmetic even if floating point
RR-0191	2.2	bounds of the type definition	Fixed point model numbers should include the
RR-0566	2.2	bounds of the type definition	Fixed point model numbers should include the
RR-0393	12.2.3	Can’t get direct visibility of	fixed point mult and div operator by renaming
AI-00262	2.2.8	Real literals with	fixed point multiplication and division
RR-0204	2.1	Clarify which	fixed point operators are predefined
AI-00521	13.3		Fixed point subtypes should not inherit SMALL
RR-0252D	2.2	the range definition	Fixed point type should include the bounds of
RR-0256	13.1	not what is needed	Fixed-point approach with range and delta is
RR-0357	10.1	Need packed decimal, wide-ranging	fixed-point, decimal deltas
RR-0591	2.2.8	Allow	fixed-point multiply/divide with universal real operands
RR-0401	2.2	efficiently because of accuracy/ Mixed-base	fixed-point operations cannot be done
RR-0744	2.2.12	Allow for loop to have non-discrete (	fixed-point) parameter
RR-0733	13.5	Need	fixed-point types not centered on zero
RR-0244B	2.3		Flag run-time errors at compile-time when possible
RR-0061	2.4	Make Long_	Float and Short_Float required types
RR-0537	13.6	Separate integer divide and	floating divide as in Pascal
RR-0144	13.3	Require support for fixed point arithmetic even if	floating point hardware is not present
AI-00609	11.1	fully characterize machine characteristics	Floating point machine attributes inadequate to
RR-0731	11.1	Arithmetic Standard as a basis for Ada’s	floating point model /the Language Compatible
RR-0252E	11.1	machine architecture Provide a	floating point model that reflects actual
RR-0255	11.1	Provide a function for returning the value of the next	floating point number
RR-0346	11.1	Need portable way to extract mantissa/exponent from	floating point number
RR-0024	11.1	Need a way to decompose	floating point numbers into mantissa/exponent
RR-0636	11.1	Improve Ada’s axioms for	floating point operations
RR-0252C	11.1	Ensure programmer can choose appropriate	floating point representation
RR-0225	11.1	accuracy is used Ensure	floating point representation with desired
RR-0564	11.1	/implementation freedom to include more mantissa digits in	floating point safe numbers
RR-0252A	11.1	Ensure support for IEEE	floating point standard; allow full use of machine/
RR-0369	11.1	Provide support for	floating point standard IEEE-754
RR-0492	11.1	Decouple mantissa and exponent information in	floating point type definitions
AI-00291	4.4	Can’t define a generic package that works for all	floating point types
AI-00518	13.4	Fixed and	floating type declarations needlessly different
RR-0189	11.1	Standard should include a	floating-point math library interface
RR-0720	11.1	hardware architectures	Floating-point model should reflect actual
RR-0664	12.3.1	Need ’IMAGE and ’VALUE attributes for	floating-point types
RR-0358	11.1	Need support for	floor, ceiling, truncate, and whole operations
RR-0535	11.1	Provide CEILING and	FLOOR numeric operators
RR-0408	4.4	There is a need for generic	formal entries
RR-0488	4.4	Allow generic	formal entries as well as generic formal subprograms
RR-0584	4.4	is given Need stricter checking of	formal generic subtypes when an instantiation
RR-0486	4.4	Allow generic formal task types as well as generic	formal limited types
RR-0375	13.7	Include	formal memory protection/security
RR-0659	4.4	Need to make entry call on a generic	formal parameter

RR-0395	2.2	Include	formal parameter names in parameter/result-type profile
RR-0600	2.2	Allow	formal parameter names in parameter/result-type profile
AI-00452	4.4	Allow record types as generic	formal parameters
AI-00478	12.3.10	Allow reading of OUT	formal parameters
RR-0714	12.3.8	Allow default names for all generic	formal parameters
AI-00451	4.4	Task entries as	formal parameters to generics
AI-00404	2.2	Use of incomplete private types in generic	formal part
RR-0462	12.3.7	Allow selected component form of type mark in a	formal part even when the selected component/
RR-0579	12.3.7	Allow a type mark of form P.FOO in the	formal part of a subprogram named FOO
RR-0722	4.4	Need generic	formal record types
RR-0169	13.4	Allow ‘‘null’’ procedures for actual or default generic	formal subprogram values
RR-0468	4.4	way to handle exceptions raised by generic	formal subprograms No generic
RR-0488	4.4	Allow generic formal entries as well as generic	formal subprograms
RR-0486	4.4	limited types Allow generic	formal task types as well as generic formal
RR-0627	4.4	Allow partial match to	formal type for records
RR-0006	4.4	Distinguish unconstrained/constrained generic	formal types
RR-0472	4.4	Distinguish unconstrained/constrained generic	formal types
RR-0622	2.1	should use ‘‘metatype’’ in describing generic	formal types The Standard
RR-0621B	4.4	Permit exceptions as generic	formals
RR-0445	4.4	Non-staticness of generic	formals poses problems
RR-0481	2.1	Make Ada documentation available in SGML	format
RR-0361	4.6	number of options for controlling the output	format of numbers Increase the
RR-0360	10.4	Add picture-	formatting capabilities to TEXT_IO
RR-0039	11.2	Make it easier to access	FORTRAN libraries
AI-00609	11.1	Floating point machine attributes inadequate to	fully characterize machine characteristics
RR-0207	4.6	Add TEXT_IO support with Exists	function and Append procedure
RR-0251	13.6	Invent new notations to distinguish	function call, array reference, and conversions
RR-0708	13.5	Allow infix	function calls
AI-00223	5.1	Require adequate resolution for the	function CLOCK
RR-0255	11.1	floating point number Provide a	function for returning the value of the next
RR-0347	5.2	control; allow task priority to increase as a	function of lack of service /under program
RR-0454	11.1	Need Entier	function or attribute for real types
RR-0453	11.1	numeric value Provide a special	function or attribute yielding the sign of a
RR-0026	13.4.3	Permit	function parameters to have modes IN OUT and OUT
RR-0598	13.4.3	Permit	function parameters to have modes OUT and IN OUT
RR-0427	12.1.1	Do not permit a	function to return a locally-declared task object
RR-0629	4.1	Need procedure and	function types for use in subprogram calls
RR-0597	4.6	Need	functional version of GET_LINE instead of procedural
RR-0047	4.6	Add TEXT_IO.GET	functions
RR-0051C	10.4	Provide packages for string edit	functions
RR-0063	5.3	from being aborted while performing critical	functions Protect tasks
RR-0130	4.6	Replace DEFAULT_xy variables in Chapter 14 by	functions
RR-0159	4.6	Add standard package of general file system	functions
RR-0620	13.6	Ban RETURN statement except inside	functions
RR-0674	13.4.1	Allow user-defined attributes as	functions
RR-0774B	13.6	Tasking defined as a standard package of	functions
RR-0489	13.1.1	Allow machine-code insertions in	functions as well as procedures
RR-0691	13.1.1	Allow machine-code insertions in	functions as well as procedures
RR-0348	11.1	Need predefined	functions for real numbers, e.g., trig, log, etc
RR-0452	13.4	overloadable constants) Allow constant	functions in static expressions (or
RR-0700	13.1	Ensure that constant	functions like sin(10.0) are evaluated at compile-time
RR-0020	5.2	so priorities should be/ Relative importance of	functions may change during program execution,
RR-0719	11.1	Need standard for trig	functions, sqrt, etc
RR-0524	6.4	objects; allow programmer to ensure pass/ Allow	functions to return references to components of
RR-0476	13.6	Allow user-written type-conversion	functions with the same name as the target type
RR-0439	4.2	Require automatic	garbage collection
RR-0643	4.2	encourage its use	Garbage collection can now be done well;
RR-0482	4.3	Multiple derived types from same package do not	generate needed operations
RR-0710	6.3	task entries to asynchronous external events	generated by operating system Need to tie
RR-0585	4.4	Need pragma to specify code-	generation strategy for generic instantiation
RR-0027	4.4	Improve generics so a generic report	generator could be written
RR-0497	13.7	/of default discriminants for types used as	generic actual can yield a surprising run-time/
RR-0342	4.4	Do not implement requests that will break	generic code sharing
RR-0693	2.2	Parameter passing rules for scalars makes	generic code sharing hard
RR-0383	4.4	Need	generic exceptions for truly reusable generic units
RR-0408	4.4	There is a need for	generic formal entries
RR-0488	4.4	formal subprograms Allow	generic formal entries as well as generic
RR-0486	4.4	Allow generic formal task types as well as	generic formal limited types

RR-0659 4.4 Need to make entry call on a generic formal parameter
AI-00452 4.4 Allow record types as generic formal parameters
RR-0714 12.3.8 Allow default names for all generic formal parameters
AI-00404 2.2 Use of incomplete private types in generic formal part
RR-0722 4.4 Need generic formal record types
RR-0169 13.4 Allow ‘‘null’’ procedures for actual or default generic formal subprogram values
RR-0468 4.4 No generic way to handle exceptions raised by generic formal subprograms
RR-0488 4.4 Allow generic formal entries as well as generic formal subprograms
RR-0486 4.4 formal limited types Allow generic formal task types as well as generic
RR-0006 4.4 Distinguish unconstrained/constrained generic formal types
RR-0472 4.4 Distinguish unconstrained/constrained generic formal types
RR-0622 2.1 The Standard should use ‘‘metatype’’ in describing generic formal types
RR-0621B 4.4 Permit exceptions as generic formals
RR-0445 4.4 Non-staticness of generic formals poses problems
RR-0055 12.4.1 Allow a subprogram body to be defined by renaming or generic instantiation
RR-0364 12.4.1 Allow a subprogram body to be defined by generic instantiation
RR-0550 12.4.1 Allow subprogram bodies to be defined by RENAMES or generic instantiation
RR-0585 4.4 Need pragma to specify code-generation strategy for generic instantiation
RR-0666 12.4.1 Allow a subprogram body to be given by generic instantiation
RR-0470 12.4.1 Allow renaming or generic instantiation to define a subprogram body
RR-0608 13.3 Allow recursive generic instantiations
AI-00291 4.4 point types Can’t define a generic package that works for all floating
RR-0005 4.4 Exception declarations in generic packages make code sharing unnecessarily/
RR-0228 4.4 Allow generic parameterization with exceptions
RR-0227 4.4 Allow generic parameterization with static numeric quantities
RR-0505B 4.4 Allow partial match for records as generic parameters
RR-0671 4.4 Allow exceptions as generic parameters
RR-0706 4.4 Allow exceptions and packages as generic parameters
RR-0774J 4.4 Allow generic parameters for any Ada entity, e.g., exceptions
RR-0027 4.4 Improve generics so a generic report generator could be written
RR-0562 4.4 Require separate compilation of generic specifications and bodies
AI-00382 2.2 Allow generic subprogram bodies
RR-0606 13.5.4 Allow generic subprogram names to be overloaded
RR-0426B 2.2 Allow declaration and body to be combined for generic subprograms
RR-0547 2.2 allow merge of specification/body for generic subprograms /non-generic subprograms,
RR-0604 2.2 allow merge of specification/body for generic subprograms /non-generic subprograms,
RR-0547 2.2 specification/body for generic/ Like non- generic subprograms, allow merge of
RR-0604 2.2 specification/body for generic/ Like non- generic subprograms, allow merge of
RR-0584 4.4 Need stricter checking of formal generic subtypes when an instantiation is given
RR-0484 4.6 Add DEFAULT_xy functionality as parameters to generic TEXT_IO packages
RR-0446 4.4 by distinguishing constrained/unconstrained generic types Tighten the contract model
RR-0190 4.4 Allow use of a base type within a generic unit
RR-0511 4.4 Allow use of a base type within a generic unit
RR-0548 13.4 Allow convenient syntax for instantiating a nested generic unit
RR-0712 4.4 declare double precision numeric types within a generic unit Need ability to
RR-0483 12.3.7 /subprogram to have the same identifier as the generic unit (as is allowed for package/
RR-0586 4.4 Different instantiations of the same generic unit may have to evaluate their actual/
RR-0033B 4.4 Need to pass exceptions to subprograms and generic units
RR-0383 4.4 Need generic exceptions for truly reusable generic units
RR-0101B 4.4 Need to pass exceptions as parameters to generic units and subprograms
RR-0526B 4.4 Need to pass exceptions as parameters to generic units and subprograms
RR-0035 13.5.4 Allow generic units to be overloaded
RR-0468 4.4 generic formal subprograms No generic way to handle exceptions raised by
RR-0174 4.3 Allow packages to be generic with respect to concurrency protection
AI-00451 4.4 Task entries as formal parameters to generics
RR-0713 4.4 Relax array matching rules for generics
RR-0027 4.4 Improve generics so a generic report generator could be written
RR-0283 4.3 Need convenient way to set global compilation parameters
RR-0618 13.1 Ban GOTO statement
RR-0116 5.2 User-modifiable priorities needed for mode change and graceful degradation
RR-0192 5.2 to change priorities during mode change and for graceful degradation Need ability
RR-0300 13.2 Use an LR grammar to define the syntax of the language
RR-0148 3.1 Provide support for extended and graphic characters (256 ASCII set)
RR-0746 13.7 Allow pictures/ graphics as comments in source code
RR-0390 3.1 Need 8-bit unsigned CHARACTER for Greek and graphics symbols
RR-0101A 4.5 Allow exceptions to be grouped under a single name
RR-0526A 4.5 Allow exceptions to be grouped under a single name
RR-0036 4.5 exception subtypes Allow exceptions to be grouped under a single name by allowing

RR-0468 4.4 No generic way to handle exceptions raised by generic formal subprograms
RR-0286D 6.3 Interrupts should be handled with a procedure model, not a task model
RR-0646 13.2 to be parameterized with parameters read in handler Allow exceptions
RR-0765 13.1 Allow ‘‘when Package_Name.others =>’’ as exception handler
RR-0772 4.5 Need to be able to get exception name in a handler
RR-0774F 2.2 Allow aliased exceptions within the same exception handler
RR-0774G 4.5 Provide exception name in OTHERS handler
RR-0145 4.5 Provide a way to get exception name from WHEN OTHERS handlers
RR-0221 13.4 Need to write common code for group of exception handlers
RR-0499 12.3.2 Like other ‘‘blocks’’, allow exception handlers in accept statements
RR-0316 6.3 Improve interrupt handling, e.g., with interrupt procedures
RR-0115 6.3 Provide better interrupt handling model
RR-0738 7.3 Add facilities to support vector processing hardware
RR-0107 5.1 Allow application to specify clock timing interval if hardware allows this flexibility
RR-0720 11.1 Floating-point model should reflect actual hardware architectures
RR-0144 13.3 fixed point arithmetic even if floating point hardware is not present Require support for
RR-0087 6.3 Allow software priorities to match/exceed hardware priorities
AI-00570 12.1.1 Releasing heap storage associated with task type instances
RR-0702 4.2 There is a need for improvements in heap storage management
RR-0109 8.1 Ada program Provide Ada semantics that are helpful when dealing with a single distributed
RR-0071 13.2 Improve support for heterogeneous distributed processing
RR-0372 13.2 Solve problem where heterogeneous processors view memory differently
RR-0558 13.4 Deriver of type should be able to hide subset of derived operations
RR-0229 13.4 value of an object to ensure these/ Need to hide the range of a scalar type and the initial
RR-0282 13.3 Ada program structure hides important context information
RR-0588 13.4 Provide a form of USE clause that hides outer homographs
RR-0402 4.3 Need unique hierarchical pathnames for subunit
RR-0442 4.3 Extend Ada to allow a package type hierarchy
RR-0588 13.4 Provide a form of USE clause that hides outer homographs
AI-00211 13.4 Additional control statement to hop to end of the loop
RR-0500 2.1 More terms should be hyphenated to improve clarity
RR-0483 12.3.7 /an instantiated subprogram to have the same identifier as the generic unit (as is allowed/
RR-0462 12.3.7 even when the selected component has the same identifier as the subprogram /in a formal part
RR-0380 7.2 Need a task identifier for every task
RR-0675 12.3.7 Allow a subprogram identifier to be used as a type mark in its specification
RR-0330 3.1 Allow national characters in literals, comments, and identifiers
RR-0707 2.2.6 Need same-name component identifiers in different variants
RR-0252A 11.1 machine characteristics Ensure support for IEEE floating point standard; allow full use of
RR-0369 11.1 Provide support for floating point standard IEEE-754
RR-0664 12.3.1 Need ’ IMAGE and ’VALUE attributes for floating-point types
RR-0495 13.6 Remove leading space in the result of the ’ IMAGE attribute for integers
RR-0363 12.3.1 discrete types Allow ’VALUE and ’ IMAGE to apply to real types as well as
RR-0602 met Encourage implementors to support standardized libraries
RR-0692 2.3 /to cause unsuccessful compilation if restrictions implied by the pragmas are not obeyed
RR-0455 4.3 The import and export mechanisms of Ada are too limited
RR-0172 4.3 Make import and export of types easier
RR-0020 5.2 program execution, so priorities/ Relative importance of functions may change during
RR-0282 13.3 Ada program structure hides important context information
RR-0421D 6.3 /timed, or conditional calls may depend inappropriately on the run-time system
RR-0690 12.4.2 subtype declaration Allow incomplete and private types to be completed by
AI-00327 2.2.5 Instantiating with an incomplete private type
AI-00404 2.2 Use of incomplete private types in generic formal part
RR-0259 13.7 Incomplete type declarations are dangerous and unnecess/
RR-0098 13.4 or private Generalize incomplete typing for types other than access
RR-0577 2.2 /constant of composite type having a component of an incompletely declared private type
RR-0743 2.2.12 Need to allow increment of something other than one in for loops
RR-0011 13.6 Expression 0**0 should not be 1 as this is an indeterminate form
RR-0133 7.2 Allow a task component of an array to get its index
RR-0571A 12.2.5 /use of OTHERS choice with named associations when index bounds are determined by context
RR-0571B 2.1 is outside the range of the applicable index constraint /the choice in an aggregate
RR-0426C 13.6 programmer errors Omitting index constraint in constant arrays causes
RR-0029 12.2.5 /use of OTHERS with named associations when the index constraint is determined by context
RR-0426D 13.4 Optional index in ’FIRST (and others) causes problems
RR-0749 12.3.11 parameters and as values in/ Should allow index sliding for slices serving as actual
RR-0755 2.2.13 Allow ‘‘[’’ instead of ‘‘(’’ for indexed components
RR-0122 2.2 to reject some integer types as array indexes Permit an implementation
RR-0510 2.2.10 Re- indexing arrays via type conversions
RR-0315 2.4 to improve/ Allow integer type names that indicate representation size, e.g., INTEGER_32,

RR-0573 12.3.11 component initialization and as/ Slide indices of array aggregates for record
RR-0515 4.2 especially in/ Need ability to request indivisible update for specific objects,
RR-0544 4.2 Need indivisible update on reference counts
RR-0708 13.5 Allow infix function calls
RR-0057 12.2.3 Need direct visibility to infix operators in another package
AI-00521 13.3 Fixed point subtypes should not inherit SMALL
RR-0525 4.3 Extend Ada to allow for polymorphism and inheritance
RR-0599 4.3 Certain changes to derived/private types will help inheritance
RR-0750 4.3 Add support for inheritance and polymorphism to the language
RR-0193 5.2 Allow priority queues, priority inheritance, and prioritized treatment of open select/
RR-0072 5.2 Prioritized queues and priority inheritance are needed for real-time applications
RR-0662 4.3 Need package classes and inheritance for object-oriented programming
RR-0021 5.2 Need priority inheritance for server tasks
RR-0125 4.3 Introduce object-oriented inheritance into the language
RR-0415 5.2 prioritized selective wait Allow priority inheritance, prioritized entry-queues, and
RR-0223 4.3 Need to add inheritance to support object-oriented programming
RR-0567 2.2 Allow variable declaration to get constraints from initial value
RR-0229 13.4 Need to hide the range of a scalar type and the initial value of an object to ensure these/
RR-0350 2.1 Clarify wording dealing with default initial values
RR-0573 12.3.11 /of array aggregates for record component initialization and as components of record/
RR-0677 2.2.2 Allow initialization clauses on scalar type declarations
RR-0595 2.2.2 Allow default initialization for all types
RR-0649 2.2.2 Allow default initialization for all types (not just records)
RR-0161 2.2.2 Allow default initialization for any non-limited type
RR-0639 8.2 Need compile-time initialization of complex data structures
RR-0456 2.2.2 Allow initialization to be associated with a type definition
RR-0506 2.2.2 Allow initialization to be associated with a type definition
RR-0230 2.2.2 Allow initialization to be associated with any type definition
RR-0129 2.2.2 non-limited type Allow default initialization to be specified for any
RR-0123 7.2 Provide initialization values to tasks at startup
RR-0086 13.4 the record itself Need to initialize a record component to the address of
AI-00479 12.3.10 Initialize access type OUT parameters to null
RR-0247 13.6 Don’t initialize access variables by default to NULL
RR-0559 13.6 If allow reading of OUT parameters, initialize OUT access to NULL
RR-0291 6.4 use of an address clause causes storage to be initialized Clarify whether
RR-0246 8.2 /are not elaborated at run time when initialized with static expressions
RR-0208 13.4 operations without waiting for/ Need ability to initiate TEXT_IO, DIRECT_IO, and SEQ_IO
RR-0398 2.2.9 Need clearer/more selective rules for pragma INLINE applicability
RR-0284 13.1.1 Machine-code insertions are unreadable; replace with INLINE macros
RR-0687 2.2.9 Pragma INLINE should not apply to all overloads; only closest
RR-0740 2.2 For optimization with respect to inlined subprograms, allow merging of scopes
RR-0575 2.2.9 Need better (more selective) control over inlining
RR-0060 2.2.9 call sites Allow inlining of subprograms from some but not all
RR-0554 9.1 for target of Unchecked_Conversion and I/O input Need constraint checks
AI-00485 4.6 interactive I/O Having independent standard input and output files is not useful for
RR-0485 4.6 Provide means to get the line length of an input or output device
RR-0552 13.4 Need ‘‘padded’’ line input with truncation and pad-fill to ’LENGTH
RR-0235 4.6 Need support for interactive terminal input/output
RR-0149 4.6 Provide a keyboard input/output package
AI-00570 12.1.1 Releasing heap storage associated with task type instances
RR-0483 12.3.7 identifier as the generic unit (as is/ Allow an instantiated subprogram to have the same
RR-0548 13.4 Allow convenient syntax for instantiating a nested generic unit
RR-0732 2.4 Clarify semantics of instantiating ENUMERATION_IO with an integer type
AI-00327 2.2.5 Instantiating with an incomplete private type
RR-0055 12.4.1 body to be defined by renaming or generic instantiation Allow a subprogram
RR-0364 12.4.1 Allow a subprogram body to be defined by generic instantiation
RR-0424 13.6 from an instance to be redefined during instantiation Allow names exported
RR-0550 12.4.1 bodies to be defined by RENAMES or generic instantiation Allow subprogram
RR-0585 4.4 to specify code-generation strategy for generic instantiation Need pragma
RR-0666 12.4.1 Allow a subprogram body to be given by generic instantiation
RR-0584 4.4 checking of formal generic subtypes when an instantiation is given Need stricter
RR-0470 12.4.1 Allow renaming or generic instantiation to define a subprogram body
RR-0608 13.3 Allow recursive generic instantiations
RR-0586 4.4 have to evaluate their actual/ Different instantiations of the same generic unit may
RR-0530 13.3 Insufficient support for mutants of limited types
RR-0409 2.4 Define in the language how 3.5 rounds to integer
RR-0635 13.4 Provide basic support for extended precision integer arithmetic
RR-0332 6.1 Provide unsigned integer capability

RR-0537	13.6	Separate	integer divide and floating divide as in Pascal
RR-0045	2.4	Allow/require extended precision for intermediate	integer results
RR-0732	2.4	of instantiating ENUMERATION_IO with an	integer type Clarify semantics
RR-0680	13.1	Predefined exponentiation should take any	integer type for exponent
RR-0315	2.4	size, e.g., INTEGER_32, to improve/ Allow	integer type names that indicate representation
RR-0188	6.1	integers and bit-wise logical operations on	integer types /applications need unsigned
RR-0433	6.1	There is a need for predefined unsigned	integer types
RR-0460	6.1	Ada needs to provide support for unsigned	integer types
RR-0572	13.1	operators with respect to all predefined	integer types Need predefined
RR-0122	2.2	Permit an implementation to reject some	integer types as array indexes
RR-0315	2.4	/type names that indicate representation size, e.g.,	INTEGER_32, to improve portability
RR-0138	6.1	Need full-sized unsigned	integers
RR-0495	13.6	space in the result of the 'IMAGE attribute for	integers Remove leading
RR-0633	6.1	Provide logical operations (e.g., XOR) for	integers
RR-0634	6.1	Provide arithmetic shift operations for	integers
RR-0188	6.1	Embedded applications need unsigned	integers and bit-wise logical operations on integer/
RR-0766	6.1	Allow bit-wise operations (AND, SHIFT) on	integers, bytes, etc
AI-00600	6.1	Why we need unsigned	integers in Ada
RR-0044	13.2	There is no need to add unsigned	integers to Ada
RR-0721	6.1	Try to add unsigned	integers to the language
AI-00460	13.1	Allow non-	integral powers for exponentiation
AI-00485	4.6	input and output files is not useful for	interactive I/O Having independent standard
AI-00488	4.6	terminators in GET routines causes problems in	interactive I/O Skipping of leading line
RR-0235	4.6	Need support for	interactive terminal input/output
RR-0074	5.2	Define a standard run-time support environment	interface
RR-0189	11.1	Standard should include a floating-point math library	interface
RR-0527	4.1	Standardize information/conventions used for pragma	INTERFACE
RR-0177	4.3	configuration management Standardize	interface between compiler and library for
RR-0175	5.2	run-time system aspects Define	interface between compiler- and target-specific
RR-0582	4.5	implementation-dependent info/ Provide standard	interface for getting additional
RR-0479	13.1	Need standard subprograms to get user-	interface information from OS
RR-0162	13.1	Provide a clean	interface to a SORT package
RR-0345	13.1	Need standardized	interface to other ANSI languages
RR-0774L	13.1	Allow pragma	INTERFACE within a package body
RR-0254	9.1	/much freedom is allowed with respect to exceptions and	intermediate expression results
RR-0045	2.4	Allow/require extended precision for	intermediate integer results
RR-0135	13.4	Catenation should not raise CONSTRAINT_ERROR for	intermediate results
RR-0220	2.2.14	Need way to get the	internal code associated with enumeration values
RR-0040	2.2.14	Need a way to determine the	internal coding of enumeration values
RR-0459	2.4	Improve support for	interoperability; lessen implementation dependence
RR-0421A	6.3	Need to delay in processing an	interrupt
RR-0195	6.3	Need	interrupt address per task, not task type
RR-0421B	6.3	different from memory address structure; a/	Interrupt address structure is sometimes
RR-0349	6.3	conceptually different and should not be/	Interrupt addresses and memory addresses are
RR-0768	5.3	Need to asynchronously	interrupt another task to stop it
RR-0735	6.3	Need ability to change	interrupt bindings at run-time
RR-0316	6.3	Improve	interrupt handling, e.g., with interrupt procedures
RR-0115	6.3	Provide better	interrupt handling model
RR-0316	6.3	Improve interrupt handling, e.g., with	interrupt procedures
RR-0151	6.3	Need standard support for priority	interrupts
RR-0421D	6.3	calls may depend/ The treatment of	interrupts as ordinary, timed, or conditional
RR-0686	6.3	Priority of	interrupts higher than normal tasks is ill-conceived
RR-0179	6.3	The treatment of	interrupts is too implementation-dependent
RR-0286D	6.3	model, not a task model	Interrupts should be handled with a procedure
RR-0286B	5.2	Embedded system user may need access to	interrupts that are also used by the run-time system
RR-0421C	6.3	task types Need to associate	interrupts with entries of task objects, not
RR-0665C	5.4	Support message-driven	intertask communication
RR-0183	5.4	Asynchronous	inter-task communication is not available
RR-0107	5.1	Allow application to specify clock timing	interval if hardware allows this flexibility
RR-0275	2.2	Error-prone and counter-	intuitive aspects of RENAMES
RR-0251	13.6	call, array reference, and conversions	Invent new notations to distinguish function
AI-00329	4.6	Look-ahead operation for TEXT_	IO
ZRR-0164	4.6	Provide multitasking terminal I/O in TEXT_	IO
RR-0360	10.4	Add picture-formatting capabilities to TEXT_	IO
RR-0593	4.6	of variant record I/O in DIRECT_IO/SEQUENTIAL_	IO Mandate implementation
RR-0626	6.2	compilers, even/ Files produced by SEQUENTIAL_	IO and DIRECT_IO are not portable among
RR-0208	13.4	Need ability to initiate TEXT_IO, DIRECT_	IO, and SEQ_IO operations without waiting for/
RR-0626	6.2	Files produced by SEQUENTIAL_IO and DIRECT_	IO are not portable among compilers, even for/

RR-0208	13.4	waiting for/ Need ability to initiate TEXT_	IO, DIRECT_IO, and SEQ_IO operations without
RR-0333	13.3	More precise definition of TEXT_	IO is needed, less implementation freedom
RR-0208	13.4	/to initiate TEXT_IO, DIRECT_IO, and SEQ_	IO operations without waiting for completion
RR-0484	4.6	functionality as parameters to generic TEXT_	IO packages Add DEFAULT_xy
RR-0207	4.6	Add TEXT_	IO support with Exists function and Append procedure
RR-0297	13.4	LOW_LEVEL_	IO was a bad idea; remove this package from the language
RR-0732	2.4	Clarify semantics of instantiating ENUMERATION_	IO with an integer type
RR-0551	4.6	Need assignment capability for TEXT_	IO.FILE_TYPE
RR-0047	4.6	Add TEXT_	IO.GET functions
RR-0295	4.6	like PUT) Create TEXT_	IO.PUT_LINE for types other than string (make
RR-0593	4.6	Mandate implementation of variant record I/O in DIRECT_	IO/SEQUENTIAL_IO
RR-0147	13.1	Add support for	ISAM
RR-0034	3.1	Ada should use	ISO 8859/1-9 (8-bit) character set
AI-00510	3.1	Use ISO symbols and standards in the Ada	ISO Standard
RR-0086	13.4	a record component to the address of the record	itself Need to initialize
RR-0271	13.6	Distinguish storage classes for variables with	key words like CONTROLLED or STATIC
RR-0149	4.6	Provide a	keyboard input/output package
RR-0397	13.6	Replace	keyword PRAGMA with something capturing meaning/
RR-0642	13.4	Add	label variables to support use of finite state machines
RR-0347	5.2	task priority to increase as a function of	lack of service /under program control; allow
RR-0113	4.2	Ensure that there are no storage	‘‘leaks’’
RR-0549	4.4	Ensure the use of unconstrained actual types is always	legal
RR-0552	13.4	line input with truncation and pad-fill to ’	LENGTH Need ‘‘padded’’
RR-0699	13.3	Do not treat an unaccepted	length clause for a type as an error
RR-0417	6.2	number of bits	Length clause should force allocation of EXACT
RR-0485	4.6	Provide means to get the line	length of an input or output device
RR-0773	6.2	Need to pack variable-	length records into a block for data transmission
RR-0054	13.2	Do not add variable	length strings to the language
RR-0327	10.4	Add varying	length strings to the language
RR-0419	10.4	Add some form of support for varying	length strings to the language
RR-0163	10.4	Need support for variable-	length strings with appropriate equality and assignment/
RR-0444	13.4	exception can be raised	Let the user limit the places where a given
RR-0039	11.2	Make it easier to access FORTRAN	libraries
RR-0178	4.3	Problems with name clashes with big program	libraries
RR-0602	met	Encourage implementors to support standardized	libraries
RR-0308	11.1	Add	libraries for array processing
RR-0073	4.3	of names to be restricted within a program	library Allow visibility
RR-0774D	13.1	Allow overloaded names in the	library
RR-0368B	4.3	those provided by the compiler/ Ensure the	library can be manipulated by tools other than
RR-0177	4.3	Standardize interface between compiler and	library for configuration management
RR-0189	11.1	Standard should include a floating-point math	library interface
RR-0370B	8.2	Can’t restart	library level tasks
RR-0370C	2.1		Library level tasks can’t terminate
RR-0226	4.3	Need standardized support for improved	library management capabilities
RR-0237	4.3	Make separate compilation independent of a particular	library model
RR-0215	2.1	Clarify termination of tasks dependent on	library packages
RR-0023	2.1	Require TERMINATE alternative to terminate	library tasks
RR-0041	4.3	subunits with respect to a common ancestor	library unit Allow overloaded
RR-0581C	13.4	name given in the context clause of a parent	library unit /a subunit to mention a package
RR-0396	12.2.1	reduce need for pragma ELABORATE Add	library unit elaboration ordering rules to
RR-0774C	4.3	Extend control of	library unit visibility
RR-0457	4.3	Structure library units as groups, control visibility of	library units
RR-0496	2.1	Clarify termination of tasks whose masters are	library units
RR-0457	4.3	library units Structure	library units as groups, control visibility of
RR-0370A	8.2	Can’t recover space declared in	library units when reconfiguring a system
RR-0601	2.2	Allow	library-level declarations to be defined by RENAMES
RR-0218	12.2.1	Make the implementation find a good	library-unit elaboration order
RR-0747	13.6	Provide better support for	‘‘light-weight’’ parallelism (as in Linda)
RR-0455	4.3	The import and export mechanisms of Ada are too	limited
RR-0670	13.6	Decouple = and /=; do not distinguish private from	limited private
RR-0184	4.2	Need user-defined assignment operator for	limited private type
RR-0578	2.2.3	Out-mode parameters of	limited private types should be allowed
RR-0129	2.2.2	initialization to be specified for any non-	limited type Allow default
RR-0161	2.2.2	Allow default initialization for any non-	limited type
RR-0392	13.5	Need ‘‘semi-’’	limited type with predefined := but no predefined =
RR-0070	4.2	Allow user-defined assignment for	limited types
RR-0160	4.2	Allow user-defined assignment for	limited types
RR-0412	12.3.9	Allow overloaded = for all types, not just	limited types

RR-0486 4.4 formal task types as well as generic formal limited types Allow generic
RR-0530 13.3 Insufficient support for mutants of limited types
RR-0272 13.6 Limited types are of little true value
RR-0001 4.2 Limited types need assignment, constants
RR-0202 4.2 Relax parameter mode rules for limited types that have an assignment operation
RR-0182 8.1 different processors Define visibility limits for parts of a program running on
RR-0747 13.6 support for ‘‘light-weight’’ parallelism (as in Linda) Provide better
RR-0553 4.6 GET_LINE should not automatically call SKIP_ LINE
RR-0355 2.4 Standardize means of getting the OS command line arguments
RR-0295 4.6 Create TEXT_IO.PUT_ LINE for types other than string (make like PUT)
RR-0552 13.4 Need ‘‘padded’’ line input with truncation and pad-fill to ’LENGTH
RR-0709 2.4 Need more portability in getting command line inputs
RR-0597 4.6 Need functional version of GET_ LINE instead of procedural
RR-0485 4.6 Provide means to get the line length of an input or output device
RR-0407A 4.5 Need exception name, line number, and unit name where raised
RR-0681 13.7 A definition of an Ada Line Of Code (LOC) should be standardized
RR-0553 4.6 GET_ LINE should not automatically call SKIP_LINE
AI-00605 4.6 which is inconsistent with other GET/ GET_ LINE skips terminators at the end of the line,
AI-00488 4.6 problems in interactive/ Skipping of leading line terminators in GET routines causes
AI-00487 4.6 not return TRUE when there is still an empty line to be read /and END_OF_FILE should
AI-00605 4.6 GET_LINE skips terminators at the end of the line, which is inconsistent with other GET/
RR-0653 8.2 declare constants whose value is supplied after linking Need to
RR-0696 13.4 Pragmas LIST and PAGE should be optional
RR-0317 2.2.12 Augment Ada’s looping: over reals, list items, etc
RR-0030 13.6 Require subprogram specification to list non-local objects referred to
RR-0096A 12.2.3 Permit renaming an enumeration literal as a character literal
RR-0156 12.3.12 A negative literal should be allowed wherever a literal is allowed
RR-0166 13.3 Allow definition of the literal representations of an abstract data type
RR-0156 12.3.12 A negative literal should be allowed wherever a literal is allowed
RR-0359 4.6 Allow mixed case output for enumeration literals
RR-0474 12.2.3 Need direct visibility to just enumeration literals and operators of a type
RR-0330 3.1 Allow national characters in literals, comments, and identifiers
RR-0302 2.4 The language should define literals for values of type ADDRESS
RR-0131 13.4 /expression, should have visibility of the enumeration literals of the qualifying type
AI-00378 12.2.3 subtype declaration Enumeration literals should be made directly visible by a
AI-00390 12.2.3 subtype declaration Character literals should be made directly visible by a
RR-0239A 12.2.3 Renaming an enumeration type should make literals visible
AI-00262 2.2.8 Real literals with fixed point multiplication and division
RR-0014 4.1 Need to call subprograms loaded in ROM
RR-0681 13.7 A definition of an Ada Line Of Code (LOC) should be standardized
RR-0427 12.1.1 Do not permit a function to return a locally-declared task object
RR-0248 13.1 record values Allow users to specify locations for discriminants that are outside
RR-0146 13.1 Support for file/record locking
RR-0348 11.1 Need predefined functions for real numbers, e.g., trig, log, etc
RR-0633 6.1 Provide logical operations (e.g., XOR) for integers
RR-0188 6.1 /need unsigned integers and bit-wise logical operations on integer types
RR-0331 3.1 Need predefined LONG_CHARACTER (16 bits) and LONG_LONG_CHA
RR-0061 2.4 Make Long_Float and Short_Float required types
RR-0331 3.1 Need predefined LONG_CHARACTER (16 bits) and LONG_LONG_CHARACTER (32)
AI-00211 13.4 Additional control statement to hop to end of the loop
RR-0305 2.1 Clarify wording of FOR loop completion
RR-0538 9.3 Create new loop structure which bans the EXIT statement
RR-0744 2.2.12 Allow for loop to have non-discrete (fixed-point) parameter
RR-0317 2.2.12 Augment Ada’s looping: over reals, list items, etc
AI-00140 12.3.12 Allow -1..10 as a discrete range in loops
RR-0717 2.2.12 Allow specification of a step size in FOR loops
RR-0743 2.2.12 increment of something other than one in for loops Need to allow
RR-0615 2.2.12 Define LOOP/UNTIL control structure as in Pascal
AI-00216 10.2 /whether characters are numeric, upper case, lower case, control, etc., independent of/
RR-0502 2.1 should be consistent in its use of upper and lower cases The Standard
RR-0297 13.4 package from the language LOW_LEVEL_IO was a bad idea; remove this
RR-0300 13.2 Use an LR grammar to define the syntax of the language
RR-0252E 11.1 Provide a floating point model that reflects actual machine architecture
AI-00609 11.1 attributes inadequate to fully characterize machine characteristics Floating point machine
RR-0252A 11.1 IEEE floating point standard; allow full use of machine characteristics Ensure support for
RR-0371 13.1.1 Need more usable and portable machine code insertions
RR-0490 2.3 /successful/convenient recovery from exceptions in machine code insertions
RR-0626 6.2 /portable among compilers, even for the same target machine e.g., because of dope vectors

RR-0284	13.1.1	with INLINE macros	Machine-code insertions are unreadable; replace
RR-0489	13.1.1	Allow	machine-code insertions in functions as well as procedures
RR-0691	13.1.1	Allow	machine-code insertions in functions as well as procedures
RR-0411	2.4	Express record representation clauses in a	machine-independent way
RR-0318	2.1	available (with embedded mark-up) Make a	machine-readable version of the Standard
RR-0642	13.4	Add label variables to support use of finite state	machines
RR-0741	7.3	Need hot performance on vector	machines; add vector types and operands
RR-0284	13.1.1	Machine-code insertions are unreadable; replace with INLINE	macros
RR-0210	13.7	Need more pragmas for software	maintenance to MIL standards
RR-0507	11.2	Provide information/control over row-	major or column-major ordering
RR-0368B	4.3	by the compiler/ Ensure the library can be	manipulated by tools other than those provided
RR-0492	11.1	point type definitions Decouple	mantissa and exponent information in floating
RR-0564	11.1	Allow implementation freedom to include more	mantissa digits in floating point safe numbers
RR-0024	11.1	Need a way to decompose floating point numbers into	mantissa/exponent
RR-0645	11.1	Need	mantissa/exponent extraction and manipulation
RR-0346	11.1	Need portable way to extract	mantissa/exponent from floating point number
RR-0062	7.1	Ensure memory	mapped devices are treated correctly by compilers
RR-0520	13.1	Language should distinguish ‘‘sequence’’ and	‘‘mapping’’ arrays
RR-0462	12.3.7	Allow selected component form of type	mark in a formal part even when the selected/
RR-0675	12.3.7	Allow a subprogram identifier to be used as a type	mark in its specification
RR-0579	12.3.7	subprogram named FOO Allow a type	mark of form P.FOO in the formal part of a
RR-0318	2.1	of the Standard available (with embedded	mark-up) Make a machine-readable version
RR-0104	12.1.1	Prohibit access to a task outside its	master
RR-0194	12.1.1	Disallow referencing a task from outside its	master
RR-0496	2.1	Clarify termination of tasks whose	masters are library units
RR-0189	11.1	Standard should include a floating-point	math library interface
RR-0354	13.6	Introduce dimensional	mathematics into the language
RR-0051A	11.1	Provide common	mathematics packages
RR-0745	13.6	Add facilities for dimensional	mathematics to the language
RR-0536	11.1	Provide MIN and	MAX numeric operators
RR-0110	6.4	access to data in different types or regions of	memory /explicit control over placement of and
RR-0238	6.4	Allow access values to designate read-only	memory
RR-0434	7.1	Need atomic read/write operations on shared volatile	memory
RR-0421B	6.3	/address structure is sometimes different from	memory address structure; a single type for/
RR-0349	6.3	should not be treated/ Interrupt addresses and	memory addresses are conceptually different and
RR-0176	9.1	Document run-time system performance and	memory allocation strategies
RR-0521	5.2	Need more convenient support for use of shared	memory among tasks
RR-0372	13.2	Solve problem where heterogeneous processors view	memory differently
RR-0541	4.2	:=, =, DESTROY operations to support	memory management Allow user-defined
RR-0374	4.2	Ada should address	memory management requirements in distributed systems
RR-0062	7.1	Ensure	memory mapped devices are treated correctly by compilers
RR-0728	8.1	Need simple Ada run-time system for distributed	memory MIMD architectures
RR-0375	13.7	Include formal	memory protection/security
RR-0150	13.7	Provide ‘‘chaining’’ of different programs to reduce	memory requirements
RR-0351	13.7	Trusted systems require auto-scrubbing of	memory when done with it
RR-0581C	13.4	Allow a pragma ELABORATE for a subunit to	mention a package name given in the context/
RR-0655	5.4	Add asynchronous	message queues
RR-0665A	5.4	Support multicast	message transfer
RR-0665C	5.4	Support	message-driven intertask communication
RR-0480	8.1	Need standard means of sending	messages between Ada programs
RR-0622	2.1	The Standard should use	‘‘metatype’’ in describing generic formal types
AI-00832	2.3	Effect of depending on parameter passing	method when calling non-Ada programs
AI-00216	10.2	numeric, upper case, lower/ Provide standard	methods for testing whether characters are
RR-0210	13.7	Need more pragmas for software maintenance to	MIL standards
RR-0728	8.1	Need simple Ada run-time system for distributed memory	MIMD architectures
RR-0536	11.1	Provide	MIN and MAX numeric operators
RR-0285	8.2		Minimize the need for run-time elaboration
RR-0359	4.6	Allow	mixed case output for enumeration literals
RR-0401	2.2	done efficiently because of accuracy/	Mixed-base fixed-point operations cannot be
RR-0286C	5.2	Run-time system should avoid entering privileged	mode
RR-0192	5.2	Need ability to change priorities during	mode change and for graceful degradation
RR-0116	5.2	User-modifiable priorities needed for	mode change and graceful degradation
AI-00003	13.1	Allow data of	mode IN in SEND_CONTROL
RR-0197	13.6	cannot be modified For access types, parameter	mode IN should mean the designated object
RR-0578	2.2.3	be allowed Out-	mode parameters of limited private types should
RR-0202	4.2	assignment operation Relax parameter	mode rules for limited types that have an
RR-0370D	5.2	Need to set priorities of tasks during	mode shifts
RR-0026	13.4.3	Permit function parameters to have	modes IN OUT and OUT

RR-0471 13.6 Allow specification of parameter modes in subprogram calls for clarity
RR-0598 13.4.3 Permit function parameters to have modes OUT and IN OUT
RR-0337 5.2 Provide some form of user- modifiable priorities
RR-0116 5.2 and graceful degradation User- modifiable priorities needed for mode change
RR-0197 13.6 IN should mean the designated object cannot be modified For access types, parameter mode
RR-0069 4.3 /subprograms and types to be added to a package without modifying the original package
RR-0393 12.2.3 Can't get direct visibility of fixed point mult and div operator by renaming
RR-0665A 5.4 Support multicast message transfer
RR-0323 13.4.2 Generalize slice for multidimensional arrays
RR-0494 13.4.2 Allow slices for any dimension in multidimensional arrays
RR-0508 13.4.2 Allow slices for any dimension in multidimensional arrays
RR-0094 2.2 Make the multiple declaration rules more complete and consistent
RR-0482 4.3 generate needed operations Multiple derived types from same package do not
RR-0052 4.3 give desired operations Multiple derived types from same package do not
RR-0377 8.2 Ada should allow partitioning of programs for multiple processor environments
RR-0289 6.2 no discriminant is present Need multiple views of a record structure even when
AI-00262 2.2.8 Real literals with fixed point multiplication and division
RR-0591 2.2.8 Allow fixed-point multiply/divide with universal real operands
RR-0164 4.6 Provide multitasking terminal I/O in TEXT_IO
RR-0530 13.3 Insufficient support for mutants of limited types
RR-0012 13.3 Mutation of types is needed for AI applications
RR-0241 5.2 Need easier and more efficient support for mutual exclusion
RR-0590 5.2 Need clear, efficient, standard support for mutual exclusion
RR-0533 4.3 packages cannot be done Mutually recursive types from different
AI-00529 13.1 Resolving the meaning of an attribute name
RR-0101A 4.5 Allow exceptions to be grouped under a single name
RR-0526A 4.5 Allow exceptions to be grouped under a single name
RR-0532 2.2.6 components in different variants to share name Allow same-type record
RR-0477 4.5 Provide a way to get the name and location of a raised exception
RR-0476 13.6 type-conversion functions with the same name as the target type Allow user-written
RR-0036 4.5 Allow exceptions to be grouped under a single name by allowing exception subtypes
RR-0178 4.3 Problems with name clashes with big program libraries
RR-0707 2.2.6 Need same- name component identifiers in different variants
AI-00582 13.4 Need a standard name for null address
RR-0145 4.5 Provide a way to get exception name from WHEN OTHERS handlers
RR-0581C 13.4 /ELABORATE for a subunit to mention a package name given in the context clause of a parent/
RR-0772 4.5 Need to be able to get exception name in a handler
RR-0774G 4.5 Provide exception name in OTHERS handler
RR-0049 13.4 Allow special notation when the same name is on both sides of :=
RR-0407A 4.5 Need exception name, line number, and unit name where raised
RR-0033A 4.5 Need to find the name of a raised exception
RR-0526C 4.5 Need to determine the name of a raised exception
RR-0085 4.5 Need to get the name of the current exception
RR-0403 4.5 Need to be able to get the name of the current exception
RR-0219 4.5 Provide a way to get the name of the last raised exception, including an out-of/
RR-0340 12.3.13 Allow optional simple name on CASE, IF, and SELECT statements
RR-0205 12.3.13 Allow program unit name on PRIVATE, BEGIN, and EXCEPTION
RR-0570 13.1 Allow the prefix of a name to denote a renaming of an enclosing construct
RR-0596 12.3.13 Allow END type_ name to substitute for END RECORD
RR-0673 12.3.13 Allow ''END RECORD type_'' name to substitute for ''END RECORD''
RR-0407A 4.5 Need exception name, line number, and unit name where raised
RR-0765 13.1 Allow ''when Package_'' Name.others => as exception handler
RR-0557 4.3 get around the inability to overload subunit names /to provide subprogram bodies helps
RR-0424 13.6 during instantiation Allow names exported from an instance to be redefined
RR-0714 12.3.8 Allow default names for all generic formal parameters
RR-0469 13.4 Parameter names for language-defined pragmas should be defined
RR-0528 2.1 Change Ada character names to recognized names for verbal communication
RR-0395 2.2 Include formal parameter names in parameter/result-type profile
RR-0600 2.2 Allow formal parameter names in parameter/result-type profile
RR-0774D 13.1 Allow overloaded names in the library
RR-0607 13.1 operator symbols Allow names of compilation units to be overloadable,
RR-0038 4.3 Allow expanded instead of simple names of subunits to be distinct
RR-0315 2.4 INTEGER_32, to improve/ Allow integer type names that indicate representation size, e.g.,
RR-0606 13.5.4 Allow generic subprogram names to be overloaded
RR-0073 4.3 Allow visibility of names to be restricted within a program library
RR-0528 2.1 Change Ada character names to recognized names for verbal communication
RR-0774H 4.5 Provide more predefined exception names with finer granularity
AI-00458 4.3 Problem with naming of subunits

RR-0050 3.1 Provide multi- national and multi-byte characters
RR-0330 3.1 Allow national characters in literals, comments, and identifiers
RR-0367 3.1 string comparison Need support for national language character sets, including
RR-0366 13.6 Subtype natural should not include zero
RR-0156 12.3.12 literal is allowed A negative literal should be allowed wherever a
RR-0637 11.1 Ada programs should run as though negative zero did not exist
RR-0548 13.4 Allow convenient syntax for instantiating a nested generic unit
AI-00214 2.2.7 Allow accept statements in program units nested in tasks
RR-0543 2.2.7 Allow accept statements in subprograms nested inside tasks
RR-0580 2.2.7 Allow accepts within subprograms/packages nested inside tasks
RR-0763 2.3 Allow nested scopes to turn off pragma SUPPRESS
RR-0568 2.2 Allow non- nested variant parts in record types
RR-0774A 4.2 Make it possible to write NEW in Ada
RR-0538 9.3 Create new loop structure which bans the EXIT statement
RR-0251 13.6 array reference, and conversions Invent new notations to distinguish function call,
RR-0540 4.3 Allow a new package to build on an existing package
RR-0322 13.1 Do not add any new reserved words to the language
RR-0255 11.1 Provide a function for returning the value of the next floating point number
RR-0250 12.1.2 Define clearer notation for expressing null ranges
RR-0049 13.4 Allow special notation when the same name is on both sides of :=
RR-0251 13.6 reference, and conversions Invent new notations to distinguish function call, array
AI-00479 12.3.10 Initialize access type OUT parameters to null
RR-0247 13.6 Don’t initialize access variables by default to NULL
RR-0559 13.6 of OUT parameters, initialize OUT access to NULL If allow reading
AI-00582 13.4 Need a standard name for null address
RR-0169 13.4 formal subprogram values Allow ‘‘null’’ procedures for actual or default generic
RR-0250 12.1.2 Define clearer notation for expressing null ranges
RR-0249 12.1.2 ’First and ’last for null ranges are defined oddly
RR-0234 12.1.2 implementation burden ‘‘Sub-’’ null ranges are of little value and an
AI-00681 13.4 Can’t declare a constant of a NULL record type
RR-0053 13.4 Allow aggregates for null records and arrays
RR-0255 11.1 returning the value of the next floating point number Provide a function for
RR-0346 11.1 extract mantissa/exponent from floating point number Need portable way to
RR-0407A 4.5 Need exception name, line number, and unit name where raised
RR-0417 6.2 Length clause should force allocation of EXACT number of bits
RR-0361 4.6 format of numbers Increase the number of options for controlling the output
RR-0127 13.4 Allow real number output in non-decimal bases
RR-0718 9.1 Need predictable results in numeric computation, especially regarding optimization
RR-0535 11.1 Provide CEILING and FLOOR numeric operators
RR-0536 11.1 Provide MIN and MAX numeric operators
RR-0227 4.4 Allow generic parameterization with static numeric quantities
AI-00285 4.4 Need to be able to access a base numeric type in some algorithms
RR-0715 2.2 Allow user-defined type conversions and attributes for numeric types
RR-0716 11.1 Unify and add attributes for numeric types
RR-0712 4.4 Need ability to declare double precision numeric types within a generic unit
AI-00216 10.2 /methods for testing whether characters are numeric, upper case, lower case, control, etc.,/
RR-0453 11.1 function or attribute yielding the sign of a numeric value Provide a special
RR-0583 2.1 Delete NUMERIC_ERROR if now subsumed by C_E
RR-0692 2.3 if restrictions implied by the pragmas are not obeyed /to cause unsuccessful compilation
RR-0287 2.4 Make access types point directly to designated object
RR-0394 13.3 Merge concepts of task and package into concept of an object
RR-0427 12.1.1 a function to return a locally-declared task object Do not permit
RR-0524 6.4 programmer to ensure pass by reference for any object /to components of objects; allow
RR-0017 6.2 Be able to treat an Ada object as an array of storage units
RR-0197 13.6 parameter mode IN should mean the designated object cannot be modified For access types,
RR-0229 13.4 /of a scalar type and the initial value of an object to ensure these values are not used/
RR-0440 4.3 Extend Ada to be truly object-oriented
RR-0125 4.3 Introduce object-oriented inheritance into the language
RR-0140 4.3 Provide support for object-oriented programming
RR-0223 4.3 Need to add inheritance to support object-oriented programming
RR-0516 4.3 Provide more support for object-oriented programming
RR-0662 4.3 Need package classes and inheritance for object-oriented programming
AI-00142 7.1 SHARED to be applied to components of composite objects Allow pragma
RR-0119 7.1 reference to elements of shared composite objects Need synchronized
RR-0258 6.4 Need access values that point to declared objects
RR-0293 6.4 Allow access values to point to declared objects
RR-0414 4.1 Ada needs subprogram types and subprogram objects
RR-0524 6.4 /functions to return references to components of objects; allow programmer to ensure pass by/

RR-0338	6.4	values and access/ Provide pointers to static	objects and safe conversion between ADDRESS
RR-0464	12.3.5	Should be able to set STORAGE_SIZE for task	objects as well as types
RR-0515	4.2	/to request indivisible update for specific	objects, especially in distributed systems
RR-0421C	6.3	Need to associate interrupts with entries of task	objects, not task types
RR-0648	12.3.5	Need to set STORAGE_SIZE on task	objects, not task types
RR-0703	12.3.5	Need to specify STORAGE_SIZE on task	objects, not task types
RR-0679	13.4	Allow component selection on	objects of a private type
RR-0430A	4.1	Need	objects of a subprogram ‘‘type’’
RR-0466	4.2	Allow user-defined finalization for	objects of a type to ensure release of resources
RR-0523	4.2	Allow user-defined finalization for	objects of a type to ensure release of resources
RR-0082	2.2.5	Allow declaration of	objects of private types in visible package specification
RR-0030	13.6	Require subprogram specification to list non-local	objects referred to
RR-0298	2.1	Clarify classes of	objects usable as attribute prefixes
RR-0013	2.1	Allow task activation to	occur at a higher priority than task execution
RR-0438	3.1	Allow use of multi-	octet character set
RR-0426C	13.6	causes programmer errors	Omitting index constraint in constant arrays
RR-0076	5.2	Allow selection of entry calls from entry queues and	open alternatives based on priorities
RR-0425	13.1	Need	open ranges in declarations of real subtypes
RR-0193	5.2	inheritance, and prioritized treatment of	open select alternatives /queues, priority
RR-0025	12.3.9	of the equality operator with different	operand types Allow overloading
RR-0591	2.2.8	Allow fixed-point multiply/divide with universal real	operands
RR-0741	7.3	on vector machines; add vector types and	operands Need hot performance
RR-0710	6.3	to asynchronous external events generated by	operating system Need to tie task entries
RR-0186	13.3	It is difficult to write an entire	operating system in Ada
RR-0201B	4.2	Overload the assignment	operation
RR-0202	4.2	rules for limited types that have an assignment	operation Relax parameter mode
AI-00329	4.6	Look-ahead	operation for TEXT_IO
RR-0052	4.3	types from same package do not give desired	operations Multiple derived
RR-0089	4.6	Provide facilities for I/O screen	operations
RR-0163	10.4	with appropriate equality and assignment	operations /support for variable-length strings
RR-0358	11.1	Need support for floor, ceiling, truncate, and whole	operations
RR-0461	5.2	Provide standard package of semaphore	operations
RR-0467	12.2.3	Need convenient way to rename a type and get its	operations
RR-0482	4.3	types from same package do not generate needed	operations Multiple derived
RR-0558	13.4	type should be able to hide subset of derived	operations Deriver of
RR-0636	11.1	Improve Ada’s axioms for floating point	operations
RR-0644	9.1	specify time bounds/constraints for certain	operations Standard should
RR-0694	12.2.3	Need easy direct visibility to the equality	operations
RR-0766	6.1	Allow bit-wise	operations (AND, SHIFT) on integers, bytes, etc
RR-0529	13.5	properties of types Allow selection of	operations based on run-time queries about
RR-0401	2.2	of accuracy/ Mixed-base fixed-point	operations cannot be done efficiently because
RR-0633	6.1	Provide logical	operations (e.g., XOR) for integers
RR-0139	6.1	Provide shift and rotate	operations for boolean arrays
RR-0634	6.1	Provide arithmetic shift	operations for integers
RR-0188	6.1	/applications need unsigned integers and bit-wise logical	operations on integer types
RR-0434	7.1	Need atomic read/write	operations on shared volatile memory
RR-0638	13.1	Axioms for built-in	operations should be specified explicitly
RR-0136	6.1	Provide support for bit-field	operations such as shift, rotate
RR-0541	4.2	Allow user-defined :=, =, DESTROY	operations to support memory management
RR-0381	2.2	Records should have composed	operations with respect to components
RR-0208	13.4	/to initiate TEXT_IO, DIRECT_IO, and SEQ_IO	operations without waiting for completion
RR-0504	13.4	Add an exchange	operator
RR-0393	12.2.3	direct visibility of fixed point mult and div	operator by renaming Can’t get
RR-0652	12.2.3	Declaring a subtype should make the equality	operator directly visible
RR-0008	12.3.9	Allow overloading of the equality	operator for all types
RR-0184	4.2	Need user-defined assignment	operator for limited private type
RR-0102	11.1	Provide explicit remainder	operator for real numbers
RR-0266	13.6		Operator overloading is dangerous
RR-0607	13.1	Allow names of compilation units to be overloadable,	operator symbols
RR-0025	12.3.9	Allow overloading of the equality	operator with different operand types
RR-0535	11.1	Provide CEILING and FLOOR numeric	operators
RR-0536	11.1	Provide MIN and MAX numeric	operators
RR-0555	12.2.3	Need ‘‘selective’’ USE clause to get just	operators and subprograms of a type
RR-0204	2.1	Clarify which fixed point	operators are predefined
RR-0022	12.2.3	Need direct visibility of	operators declared in another package
RR-0057	12.2.3	Need direct visibility to infix	operators in another package
RR-0232	12.2.3	Need to allow direct visibility of	operators in packages
RR-0474	12.2.3	visibility to just enumeration literals and	operators of a type Need direct

AI-00480 12.2.3 subtype declaration — Operators should be made directly visible by a
RR-0682 13.5.1 Allow user-defined overloaded — operators such as ‘‘?’’, ‘‘:-’’, etc
RR-0201A 13.5.1 Liberalize overloading of — operators to other character sequences
RR-0572 13.1 integer types Need predefined — operators with respect to all predefined
RR-0685 2.2 Clarify and loosen 11.6 to allow more — optimization
RR-0718 9.1 in numeric computation, especially regarding — optimization Need predictable results
RR-0739 2.2 Relax 11.6 canonical order rules to allow more — optimization
RR-0387 2.2 Relax 11.6 — optimization rules to allow compiler to do more optimizing
RR-0729 9.1 Language should provide way to turn off — optimization to eliminate bugs
RR-0740 2.2 subprograms, allow merging of scopes For — optimization with respect to inlined
RR-0386 9.1 Need standard way of telling the compiler not to — optimize
AI-00280 2.2 Allow pragma — OPTIMIZE in package specifications
RR-0387 2.2 optimization rules to allow compiler to do more — optimizing Relax 11.6
RR-0218 12.2.1 find a good library-unit elaboration — order Make the implementation
RR-0137 2.4 Standardize bit storage/ — order conventions
RR-0042 2.3 Clarify the meaning of incorrect- — order dependence and its effects
RR-0066 2.3 associated with erroneous execution/incorrect — order dependences Reduce risks
RR-0657 5.2 — Order entry queues based on priority
RR-0428 12.2.2 — Order of declarations is too restrictive
RR-0767 12.2.1 pragma ELABORATE Solve the elaboration — order problem without requiring the use of
RR-0739 2.2 Relax 11.6 canonical — order rules to allow more optimization
RR-0507 11.2 over row-major or column-major — ordering Provide information/control
RR-0396 12.2.1 Add library unit elaboration — ordering rules to reduce need for pragma ELABORATE
RR-0586 4.4 evaluate their actual parameters in different — orders /of the same generic unit may have to
RR-0421D 6.3 The treatment of interrupts as — ordinary, timed, or conditional calls may depend/
RR-0440 4.3 Extend Ada to be truly object- — oriented
RR-0125 4.3 Introduce object- — oriented inheritance into the language
RR-0140 4.3 Provide support for object- — oriented programming
RR-0223 4.3 Need to add inheritance to support object- — oriented programming
RR-0516 4.3 Provide more support for object- — oriented programming
RR-0662 4.3 Need package classes and inheritance for object- — oriented programming
RR-0069 4.3 to be added to a package without modifying the — original package Allow subprograms and types
RR-0479 13.1 to get user-interface information from — OS Need standard subprograms
RR-0355 2.4 Standardize means of getting the — OS command line arguments
RR-0426D 13.4 Optional index in ’FIRST (and — others) causes problems
RR-0571A 12.2.5 index bounds are determined by/ Allow use of — OTHERS choice with named associations when
RR-0774G 4.5 Provide exception name in — OTHERS handler
RR-0145 4.5 Provide a way to get exception name from WHEN — OTHERS handlers
RR-0605 12.2.5 Rules for — OTHERS in aggregates are confusing
RR-0029 12.2.5 constraint is determined by/ Allow use of — OTHERS with named associations when the index
RR-0026 13.4.3 Permit function parameters to have modes IN OUT and — OUT
RR-0598 13.4.3 Permit function parameters to have modes OUT and IN — OUT
RR-0559 13.6 If allow reading of OUT parameters, initialize — OUT access to NULL
RR-0598 13.4.3 Permit function parameters to have modes — OUT and IN OUT
RR-0026 13.4.3 Permit function parameters to have modes IN — OUT and OUT
RR-0103A 2.2.3 Allow unchecked conversion for IN — OUT and OUT parameters
AI-00478 12.3.10 Allow reading of — OUT formal parameters
RR-0404 4.6 Need convenient way to find — out if a particular file exists
RR-0213 2.4 Need to be able to find — out if an implementation rounds up or down
RR-0264 13.3 Discriminants need to stand — out more
AI-00840 2.2.3 Allow access — OUT parameter as attribute prefix
RR-0002 12.3.10 Allow reading of — OUT parameters
RR-0103A 2.2.3 Allow unchecked conversion for IN OUT and — OUT parameters
RR-0303 12.3.10 Allow reading of — OUT parameters
RR-0539 12.3.10 Allow reading of — OUT parameters
RR-0574 2.2 Inability to eliminate constraint check for — OUT parameters
RR-0559 13.6 If allow reading of — OUT parameters, initialize OUT access to NULL
AI-00479 12.3.10 Initialize access type — OUT parameters to null
RR-0621A 4.5 Need to find — out which exception has been raised
RR-0588 13.4 Provide a form of USE clause that hides — outer homographs
RR-0578 2.2.3 should be allowed — Out-mode parameters of limited private types
RR-0219 4.5 name of the last raised exception, including an — out-of-scope exception /a way to get the
RR-0235 4.6 Need support for interactive terminal input/ — output
RR-0485 4.6 Provide means to get the line length of an input or — output device
AI-00485 4.6 Having independent standard input and — output files is not useful for interactive I/O
RR-0359 4.6 Allow mixed case — output for enumeration literals
RR-0361 4.6 Increase the number of options for controlling the — output format of numbers
RR-0127 13.4 Allow real number — output in non-decimal bases

RR-0149	4.6	Provide a keyboard input/	output package
RR-0724	2.1	implicit conversion Need clearer/simpler	overload resolution rules, especially for
RR-0519	13.1	Simplify	overload rules for ambiguous/universal expressions
RR-0557	4.3	bodies helps get around the inability to	overload subunit names /to provide subprogram
RR-0201B	4.2		Overload the assignment operation
RR-0452	13.4	Allow constant functions in static expressions (or	overloadable constants)
RR-0429	12.2.3	Need construct that makes just	overloadable declarations directly visible
RR-0607	13.1	Allow names of compilation units to be	overloadable, operator symbols
RR-0035	13.5.4	Allow generic units to be	overloaded
RR-0606	13.5.4	Allow generic subprogram names to be	overloaded
RR-0412	12.3.9	Allow	overloaded = for all types, not just limited types
RR-0774D	13.1	Allow	overloaded names in the library
RR-0682	13.5.1	Allow user-defined	overloaded operators such as “?”, “:-”, etc
RR-0041	4.3	ancestor library unit Allow	overloaded subunits with respect to a common
RR-0266	13.6	Operator	overloading is dangerous
RR-0663	4.2	Allow certain	overloading of := and subscripting
RR-0513	12.3.9	an array type Allow	overloading of = for any type, e.g., returning
RR-0201A	13.5.1	Liberalize	overloading of operators to other character sequences
RR-0008	12.3.9	Allow	overloading of the equality operator for all types
RR-0025	12.3.9	different operand types Allow	overloading of the equality operator with
RR-0687	2.2.9	Pragma INLINE should not apply to all	overloads; only closest
RR-0730	13.4	The private part of a package should have its	own context clause
RR-0773	6.2	data transmission Need to	pack variable-length records into a block for
RR-0022	12.2.3	Need direct visibility of operators declared in another	package
RR-0057	12.2.3	Need direct visibility to infix operators in another	package
RR-0069	4.3	to a package without modifying the original	package /subprograms and types to be added
RR-0149	4.6	Provide a keyboard input/output	package
RR-0162	13.1	Provide a clean interface to a SORT	package
RR-0270	13.4	Allow specification of read-only data from a	package
RR-0540	4.3	Allow a new package to build on an existing	package
RR-0624	12.2.3	Provide selective direct visibility into a	package
RR-0093	13.4	of deferred constants to be given in a	package body Allow full declaration
RR-0307	4.3	Allow completion of private declarations to be in the	package body
RR-0774L	13.1	Allow pragma INTERFACE within a	package body
RR-0725	12.4.1	Need rename in	package body for routine in package specification
RR-0426A	12.2.4	The effect of an optional	package body is confusing to users
AI-00442	13.4	Time zone information in	package CALENDAR
RR-0662	4.3	object-oriented programming Need	package classes and inheritance for
RR-0451	4.3	Changes to	package constants should not cause recompilation
RR-0727	12.2.3	Need selective direct visibility of	package declarations
RR-0482	4.3	Multiple derived types from same	package do not generate needed operations
RR-0052	4.3	Multiple derived types from same	package do not give desired operations
RR-0297	13.4	LOW_LEVEL_IO was a bad idea; remove this	package from the language
RR-0483	12.3.7	as the generic unit (as is allowed for	package instances) /to have the same identifier
RR-0394	13.3	Merge concepts of task and	package into concept of an object
RR-0581C	13.4	/a pragma ELABORATE for a subunit to mention a	package name given in the context clause of a/
RR-0748	5.4	Provide standard	package of asynchronous primitives
RR-0774B	13.6	Tasking defined as a standard	package of functions
RR-0159	4.6	Add standard	package of general file system functions
RR-0461	5.2	Provide standard	package of semaphore operations
RR-0730	13.4	The private part of a	package should have its own context clause
RR-0082	2.2.5	of objects of private types in visible	package specification Allow declaration
RR-0725	12.4.1	Need rename in package body for routine in	package specification
AI-00280	2.2	Allow pragma OPTIMIZE in	package specifications
RR-0581B	2.1	Clarify the effect of applying pragma ELABORATE to a	package that has no body
AI-00291	4.4	Can’t define a generic	package that works for all floating point types
RR-0540	4.3	Allow a new	package to build on an existing package
RR-0442	4.3	Extend Ada to allow a	package type hierarchy
RR-0081	4.1	Provide subprogram and	package types
RR-0660	4.2	Need constructors and destructors for	package types
RR-0668	4.3	Need	package types to get, for example, an array of packages
RR-0069	4.3	Allow subprograms and types to be added to a	package without modifying the original package
RR-0765	13.1	Allow ‘‘when’’	Package_Name.others => as exception handler
RR-0051A	11.1	Provide common mathematics	packages
RR-0051B	10.4	Provide standard string manipulation	packages
RR-0215	2.1	Clarify termination of tasks dependent on library	packages
RR-0232	12.2.3	Need to allow direct visibility of operators in	packages
RR-0385	4.2	Need finalization code for	packages

RR-0478	13.1	for restricting use of resources to trusted	packages Add language facilities
RR-0484	4.6	functionality as parameters to generic TEXT_IO	packages Add DEFAULT_xy
RR-0560	4.3	a private type's representation in related	packages Need to access
RR-0668	4.3	Need package types to get, for example, an array of	packages
RR-0203	4.2	Allow finalization code for	packages and tasks
RR-0294	met	applications; make Chapter 14 optional I/O	packages are not suitable for embedded
RR-0706	4.4	Allow exceptions and	packages as generic parameters
RR-0533	4.3	Mutually recursive types from different	packages cannot be done
RR-0222	8.1	Need additional predefined	packages for process control/communication
RR-0051C	10.4	Provide	packages for string edit functions
RR-0005	4.4	Exception declarations in generic	packages make code sharing unnecessarily difficult
RR-0684	4.3	Related	packages need access to a private type's representation
RR-0580	2.2.7	Allow accepts within subprograms/	packages nested inside tasks
RR-0296	met	Make predefined I/O	packages optional if appropriate
RR-0174	4.3	concurrency protection Allow	packages to be generic with respect to
RR-0257	2.1	Ensure that BOOLEAN and BYTE arrays can be tightly	packed
RR-0357	10.1	decimal deltas Need	packed decimal, wide-ranging fixed-point,
RR-0310	10.4	Need convenient way to	pad with blanks in string assignments
RR-0552	13.4	pad-fill to 'LENGTH Need	''padded'' line input with truncation and
AI-00487	4.6	when there is still an empty line to/ END_OF_	PAGE and END_OF_FILE should not return TRUE
RR-0696	13.4	Pragmas LIST and	PAGE should be optional
RR-0625	13.6	Change EXIT/WHEN to WHEN/EXIT to	parallel Ada IF and English
RR-0665B	8.2	Support allocation of	parallel processes to processors
RR-0514	7.3	Provide support for simple	parallel threads within a program unit
RR-0747	13.6	Provide better support for ''light-weight''	parallelism (as in Linda)
RR-0376	13.3	Need special treatment of exceptions in distributed/	parallel/multi-processor systems
AI-00473	2.2.3	actual parameter should be allowed as a default	parameter Any form of
RR-0659	4.4	Need to make entry call on a generic formal	parameter
RR-0744	2.2.12	Allow for loop to have non-discrete (fixed-point)	parameter
AI-00840	2.2.3	Allow access OUT	parameter as attribute prefix
RR-0214	13.1	Require that a subprogram	parameter be used within the body
RR-0165	2.3	compile-time errors Allow	parameter constraint violations to be
RR-0576	13.4	previous IN parameters Allow	parameter default expressions to make use of
RR-0197	13.6	object cannot be modified For access types,	parameter mode IN should mean the designated
RR-0202	4.2	have an assignment operation Relax	parameter mode rules for limited types that
RR-0471	13.6	Allow specification of	parameter modes in subprogram calls for clarity
RR-0469	13.4	should be defined	Parameter names for language-defined pragmas
RR-0395	2.2	Include formal	parameter names in parameter/result-type profile
RR-0600	2.2	Allow formal	parameter names in parameter/result-type profile
RR-0217	13.1	Require that a	parameter of an entry be used within an accept
AI-00832	2.3	Effect of depending on	parameter passing method when calling non-Ada programs
RR-0693	2.2	generic code sharing hard	Parameter passing rules for scalars makes
AI-00473	2.2.3	Any form of actual	parameter should be allowed as a default parameter
RR-0239B	2.2.3	A renamed type cannot be used in an actual	parameter type conversion
RR-0097	13.4	Allow/require explicit action to get default	parameter value
RR-0228	4.4	Allow generic	parameterization with exceptions
RR-0227	4.4	Allow generic	parameterization with static numeric quantities
RR-0646	13.2	Allow exceptions to be	parameterized with parameters read in handler
RR-0395	2.2	Include formal parameter names in	parameter/result-type profile
RR-0600	2.2	Allow formal parameter names in	parameter/result-type profile
AI-00452	4.4	Allow record types as generic formal	parameters
AI-00478	12.3.10	Allow reading of OUT formal	parameters
RR-0002	12.3.10	Allow reading of OUT	parameters
RR-0103A	2.2.3	Allow unchecked conversion for IN OUT and OUT	parameters
RR-0283	4.3	Need convenient way to set global compilation	parameters
RR-0303	12.3.10	Allow reading of OUT	parameters
RR-0430B	4.1	Need to pass subprograms as	parameters
RR-0505B	4.4	Allow partial match for records as generic	parameters
RR-0539	12.3.10	Allow reading of OUT	parameters
RR-0574	2.2	Inability to eliminate constraint check for OUT	parameters
RR-0576	13.4	default expressions to make use of previous IN	parameters Allow parameter
RR-0671	4.4	Allow exceptions as generic	parameters
RR-0706	4.4	Allow exceptions and packages as generic	parameters
RR-0714	12.3.8	Allow default names for all generic formal	parameters
RR-0774K	4.1	Allow subprograms as	parameters
RR-0749	12.3.11	/index sliding for slices serving as actual	parameters and as values in record components
RR-0422	4.1	Allow subprograms as	parameters and maybe also as values
RR-0611	4.1	Allow subprogram types, variables, constants,	parameters, etc

RR-0774J 4.4 Allow generic parameters for any Ada entity, e.g., exceptions
RR-0180 4.1 There is a need for procedures as parameters for X-Windows, etc
RR-0334 7.2 to process part of an/ Need to specify task parameters giving a task its work domain, e.g.,
RR-0586 4.4 generic unit may have to evaluate their actual parameters in different orders /of the same
RR-0559 13.6 If allow reading of OUT parameters, initialize OUT access to NULL
RR-0578 2.2.3 Out-mode parameters of limited private types should be allowed
RR-0646 13.2 Allow exceptions to be parameterized with parameters read in handler
RR-0484 4.6 Add DEFAULT_xy functionality as parameters to generic TEXT_IO packages
RR-0101B 4.4 Need to pass exceptions as parameters to generic units and subprograms
RR-0526B 4.4 Need to pass exceptions as parameters to generic units and subprograms
AI-00451 4.4 Task entries as formal parameters to generics
RR-0026 13.4.3 Permit function parameters to have modes IN OUT and OUT
RR-0598 13.4.3 Permit function parameters to have modes OUT and IN OUT
AI-00479 12.3.10 Initialize access type OUT parameters to null
RR-0512 4.1 Provide subprograms as parameters to subprograms
RR-0128 4.1 Provide subprograms as parameters to subprograms and entries
RR-0641 4.1 Add subprograms as parameters to the language
RR-0556 2.2.13 the language Parentheses are used for too many purposes in
RR-0299 13.1 Make everything in the Standard ‘‘part of the standard’’
RR-0505B 4.4 Allow partial match for records as generic parameters
RR-0627 4.4 Allow partial match to formal type for records
RR-0473 13.5 discriminated records Allow ‘‘partially’’ constrained subtypes of
RR-0537 13.6 Separate integer divide and floating divide as in Pascal
RR-0615 2.2.12 Define LOOP/UNTIL control structure as in Pascal
RR-0524 6.4 of objects; allow programmer to ensure pass by reference for any object /to components
RR-0101B 4.4 and subprograms Need to pass exceptions as parameters to generic units
RR-0526B 4.4 and subprograms Need to pass exceptions as parameters to generic units
RR-0033B 4.4 Need to pass exceptions to subprograms and generic units
RR-0430B 4.1 Need to pass subprograms as parameters
AI-00832 2.3 Effect of depending on parameter passing method when calling non-Ada programs
RR-0693 2.2 sharing hard Parameter passing rules for scalars makes generic code
RR-0402 4.3 Need unique hierarchical pathnames for subunit
RR-0176 9.1 Document run-time system performance and memory allocation strategies
RR-0084 5.2 conventions for using tasks that permit high- performance implementations Specify standard
RR-0280 5.1 inefficient; Calendar time unnecessary; timing performance must be documented /delays are too
RR-0741 7.3 types and operands Need hot performance on vector machines; add vector
RR-0705 2.2 For better performance, remove restrictions on static expressions
RR-0063 5.3 Protect tasks from being aborted while performing critical functions
RR-0410 5.1 Provide explicit language support for periodic tasks
RR-0579 12.3.7 Allow a type mark of form P.FOO in the formal part of a subprogram named FOO
RR-0360 10.4 Add picture-formatting capabilities to TEXT_IO
RR-0746 13.7 Allow pictures/graphics as comments in source code
RR-0110 6.4 types or regions/ Provide explicit control over placement of and access to data in different
RR-0444 13.4 Let the user limit the places where a given exception can be raised
RR-0447 4.6 to preserve/restore the default file at any point Need to be able
RR-0256 13.1 is needed Fixed- point approach with range and delta is not what
RR-0144 13.3 hardware is not/ Require support for fixed point arithmetic even if floating point
RR-0357 10.1 Need packed decimal, wide-ranging fixed- point, decimal deltas
RR-0287 2.4 Make access types point directly to designated object
RR-0144 13.3 for fixed point arithmetic even if floating point hardware is not present Require support
RR-0436 2.1 Clarify task synchronization point inconsistencies
AI-00609 11.1 characterize machine characteristics Floating point machine attributes inadequate to fully
RR-0189 11.1 Standard should include a floating- point math library interface
RR-0731 11.1 Standard as a basis for Ada’s floating point model /the Language Compatible Arithmetic
RR-0191 2.2 of the type definition Fixed point model numbers should include the bounds
RR-0566 2.2 of the type definition Fixed point model numbers should include the bounds
RR-0720 11.1 Floating- point model should reflect actual hardware architectures
RR-0252E 11.1 Provide a floating point model that reflects actual machine architecture
RR-0393 12.2.3 Can’t get direct visibility of fixed point mult and div operator by renaming
AI-00262 2.2.8 Real literals with fixed point multiplication and division
RR-0591 2.2.8 Allow fixed- point multiply/divide with universal real operands
RR-0255 11.1 for returning the value of the next floating point number Provide a function
RR-0346 11.1 way to extract mantissa/exponent from floating point number Need portable
RR-0024 11.1 Need a way to decompose floating point numbers into mantissa/exponent
RR-0636 11.1 Improve Ada’s axioms for floating point operations
RR-0401 2.2 because of accuracy/ Mixed-base fixed- point operations cannot be done efficiently
RR-0204 2.1 Clarify which fixed point operators are predefined
RR-0744 2.2.12 Allow for loop to have non-discrete (fixed- point) parameter

RR-0252C 11.1 Ensure programmer can choose appropriate floating point representation
RR-0225 11.1 Ensure floating point representation with desired accuracy is used
RR-0564 11.1 to include more mantissa digits in floating point safe numbers /implementation freedom
RR-0252A 11.1 Ensure support for IEEE floating point standard; allow full use of machine/
RR-0369 11.1 Provide support for floating point standard IEEE-754
AI-00521 13.3 Fixed point subtypes should not inherit SMALL
RR-0258 6.4 Need access values that point to declared objects
RR-0293 6.4 Allow access values to point to declared objects
RR-0492 11.1 Decouple mantissa and exponent information in floating point type definitions
RR-0252D 2.2 range definition Fixed point type should include the bounds of the
AI-00291 4.4 a generic package that works for all floating point types Can't define
RR-0664 12.3.1 Need 'IMAGE and 'VALUE attributes for floating- point types
RR-0733 13.5 Need fixed- point types not centered on zero
RR-0672 13.4 Need anonymous pointer types
RR-0726 6.4 Need non-contiguous arrays, static pointers
RR-0338 6.4 between ADDRESS values and access/ Provide pointers to static objects and safe conversion
RR-0441 4.1 Extend Ada to allow for polymorphism
RR-0525 4.3 Extend Ada to allow for polymorphism and inheritance
RR-0750 4.3 Add support for inheritance and polymorphism to the language
RR-0253 13.1 DIGITS and DELTA approach leads to inefficiency, non- portability
RR-0315 2.4 size, e.g., INTEGER_32, to improve portability /names that indicate representation
RR-0365 2.4 variations in implementations to increase portability Reduce allowed
RR-0432 2.4 Severely limit implementation options to improve portability
RR-0709 2.4 Need more portability in getting command line inputs
RR-0613 13.4.1 User-defined attributes solve portability problems with/
RR-0124 5.2 code dependent on task scheduling algorithms is portable Ensure that
RR-0626 6.2 /produced by SEQUENTIAL_IO and DIRECT_IO are not portable among compilers, even for the same/
RR-0698 4.3 Need ability to separate portable and non-portable code into separate units
RR-0371 13.1.1 Need more usable and portable machine code insertions
RR-0043 13.1.1 Make it easier and more portable to use assembler with Ada
RR-0346 11.1 floating point number Need portable way to extract mantissa/exponent from
RR-0581 12.2.1 and unhelpful Rules specifying the position of pragma ELABORATE are error-prone
RR-0198 13.4 Allow positional aggregate for single-component aggregate
RR-0065 4.3 pragmas to be separated from/ To improve reuse possibilities, allow rep clauses and various
RR-0236 2.4 or at least, ensure it is documented whenever possible /implementation-dependent behavior,
RR-0244B 2.3 Flag run-time errors at compile-time when possible
RR-0774A 4.2 Make it possible to write NEW in Ada
AI-00460 13.1 Allow non-integral powers for exponentiation
AI-00421 12.2.1 Eliminate pragma ELABORATE
RR-0095 13.4 Allow applicable units to be named in USE clauses and pragma ELABORATE
RR-0396 12.2.1 elaboration ordering rules to reduce need for pragma ELABORATE Add library unit
RR-0767 12.2.1 order problem without requiring the use of pragma ELABORATE Solve the elaboration
RR-0581 12.2.1 Rules specifying the position of pragma ELABORATE are error-prone and unhelpful
RR-0581C 13.4 package name given in the context/ Allow a pragma ELABORATE for a subunit to mention a
RR-0546 12.2.1 It is too difficult to ensure that pragma ELABORATE is used when it is needed
RR-0581A 12.2.1 Eliminate need for pragma ELABORATE; pragma NOT_ELABORATE might
RR-0004 12.2.1 Pragma ELABORATE should be transitive
RR-0233 12.2.1 Pragma ELABORATE should be transitive
RR-0581B 2.1 Clarify the effect of applying pragma ELABORATE to a package that has no body
RR-0398 2.2.9 Need clearer/more selective rules for pragma INLINE applicability
RR-0687 2.2.9 overloads; only closest Pragma INLINE should not apply to all
RR-0527 4.1 Standardize information/conventions used for pragma INTERFACE
RR-0774L 13.1 Allow pragma INTERFACE within a package body
RR-0581A 12.2.1 Eliminate need for pragma ELABORATE; pragma NOT_ELABORATE might help
AI-00280 2.2 Allow pragma OPTIMIZE in package specifications
RR-0678 7.1 between programs; need VOLATILE Pragma SHARED is not sufficient for data shared
AI-00142 7.1 composite objects Allow pragma SHARED to be applied to components of
RR-0763 2.3 Allow nested scopes to turn off pragma SUPPRESS
RR-0585 4.4 generic instantiation Need pragma to specify code-generation strategy for
RR-0397 13.6 Replace keyword PRAGMA with something capturing meaning better
RR-0211 2.3 Require compilers to report unrecognized or incorrect pragmas
RR-0754 2.3 Require warnings for unrecognized pragmas
RR-0756 2.3 Require warnings when pragmas are ignored
RR-0692 2.3 compilation if restrictions implied by the pragmas are not obeyed /to cause unsuccessful
AI-00850 2.3 Rejecting a unit when a pragma's assumptions are not met
RR-0210 13.7 Need more pragmas for software maintenance to MIL standards
RR-0696 13.4 Pragmas LIST and PAGE should be optional
RR-0469 13.4 Parameter names for language-defined pragmas should be defined

RR-0065 4.3 /possibilities, allow rep clauses and various pragmas to be separated from the compilation/
RR-0692 2.3 Allow implementation-defined pragmas to cause unsuccessful compilation if restrictions/
RR-0276 5.1 Need user specified accuracy and precision control over timing
RR-0045 2.4 Allow/require extended precision for intermediate integer results
RR-0635 13.4 Provide basic support for extended precision integer arithmetic
RR-0712 4.4 Need ability to declare double precision numeric types within a generic unit
RR-0204 2.1 Clarify which fixed point operators are predefined
RR-0392 13.5 Need ‘‘semi-limited’’ type with predefined := but no predefined =
RR-0774H 4.5 Provide more predefined exception names with finer granularity
RR-0399 13.1 Break up overly broad predefined exceptions, e.g., CONSTRAINT_ERROR
RR-0416 4.5 Granularity of predefined exceptions is too coarse
RR-0680 13.1 integer type for exponent Predefined exponentiation should take any
RR-0348 11.1 trig, log, etc Need predefined functions for real numbers, e.g.,
RR-0572 13.1 Need predefined operators with respect to all predefined integer types
RR-0296 met Make predefined I/O packages optional if appropriate
RR-0331 3.1 LONG_LONG_CHARACTER (32) Need predefined LONG_CHARACTER (16 bits) and
RR-0572 13.1 predefined integer types Need predefined operators with respect to all
RR-0222 8.1 Need additional predefined packages for process control/communication
RR-0433 6.1 There is a need for predefined unsigned integer types
RR-0718 9.1 especially regarding optimization Need predictable results in numeric computation,
AI-00840 2.2.3 Allow access OUT parameter as attribute prefix
RR-0570 13.1 enclosing construct Allow the prefix of a name to denote a renaming of an
RR-0298 2.1 Clarify classes of objects usable as attribute prefixes
RR-0497 13.7 used as generic actual can yield a surprising/ Presence of default discriminants for types
RR-0748 5.4 Provide standard package of asynchronous primitives
RR-0076 5.2 entry queues and open alternatives based on priorities Allow selection of entry calls from
RR-0087 6.3 Allow software priorities to match/exceed hardware priorities
RR-0337 5.2 Provide some form of user-modifiable priorities
RR-0654 5.2 Need non-static priorities
RR-0192 5.2 Need ability to change priorities during mode change and for graceful degradation
RR-0116 5.2 User-modifiable priorities needed for mode change and graceful degradation
RR-0370D 5.2 Need to set priorities of tasks during mode shifts
RR-0020 5.2 /of functions may change during program execution, so priorities should be changeable
RR-0015 5.2 Allow task priorities to control all queuing/select decisions
RR-0087 6.3 Allow software priorities to match/exceed hardware priorities
RR-0347 5.2 priority to/ Allow applications to change priorities under program control; allow task
RR-0415 5.2 selective wait Allow priority inheritance, prioritized entry-queues, and prioritized
RR-0072 5.2 needed for real-time applications Prioritized queues and priority inheritance are
RR-0415 5.2 inheritance, prioritized entry-queues, and prioritized selective wait Allow priority
RR-0193 5.2 Allow priority queues, priority inheritance, and prioritized treatment of open select/
RR-0657 5.2 Order entry queues based on priority
RR-0193 5.2 of open select/ Allow priority queues, priority inheritance, and prioritized treatment
RR-0072 5.2 Prioritized queues and priority inheritance are needed for real-time applications
RR-0021 5.2 Need priority inheritance for server tasks
RR-0415 5.2 and prioritized selective wait Allow priority inheritance, prioritized entry-queues,
RR-0151 6.3 Need standard support for priority interrupts
RR-0686 6.3 is ill-conceived Priority of interrupts higher than normal tasks
RR-0075 5.2 Queue entries by task priority or FIFO based on application
RR-0193 5.2 prioritized treatment of open select/ Allow priority queues, priority inheritance, and
RR-0013 2.1 Allow task activation to occur at a higher priority than task execution
RR-0347 5.2 /priorities under program control; allow task priority to increase as a function of lack of/
RR-0098 13.4 typing for types other than access or private Generalize incomplete
RR-0670 13.6 and /=; do not distinguish private from limited private Decouple =
RR-0205 12.3.13 Allow program unit name on PRIVATE, BEGIN, and EXCEPTION
RR-0307 4.3 Allow completion of private declarations to be in the package body
RR-0670 13.6 Decouple = and /=; do not distinguish private from limited private
RR-0153 13.1 and implementation Private part foils separation of specification
RR-0730 13.4 context clause The private part of a package should have its own
RR-0628 2.2.11 Need private task entries
RR-0487 2.2.11 the task Need private task entries for exclusive use within
AI-00327 2.2.5 Instantiating with an incomplete private type
RR-0184 4.2 Need user-defined assignment operator for limited private type
RR-0577 2.2 having a component of an incompletely declared private type /constant of composite type
RR-0679 13.4 Allow component selection on objects of a private type
RR-0542 2.2.5 One way or another allow usage of private type before its completion declaration
AI-00540 12.4.2 Completing a private type declaration with a subtype declaration
RR-0096C 12.4.2 Allow the full declaration of a private type to be provided by a renaming declaration
RR-0010 2.2 Allow the full declaration of a private type with discriminants to be a derived type

RR-0313 13.4 Allow deferred constants of arbitrary (i.e., non-private) types
RR-0423 2.2 restriction on full declarations of private types Remove discriminant
RR-0449 13.1 Do not allow unchecked conversion of private types
RR-0509 13.4.1 Allow user-defined attributes for user-defined or private types
AI-00404 2.2 Use of incomplete private types in generic formal part
RR-0273 13.6 There are problems with private types in the language
RR-0082 2.2.5 Allow declaration of objects of private types in visible package specification
RR-0684 4.3 Related packages need access to a private type's representation
RR-0560 4.3 Need to access a private type's representation in related packages
RR-0578 2.2.3 Out-mode parameters of limited private types should be allowed
RR-0690 12.4.2 Allow incomplete and private types to be completed by subtype declaration
RR-0599 4.3 Certain changes to derived/ private types will help inheritance
RR-0286C 5.2 Run-time system should avoid entering privileged mode
RR-0597 4.6 Need functional version of GET_LINE instead of procedural
RR-0207 4.6 Add TEXT_IO support with Exists function and Append procedure
RR-0629 4.1 subprogram calls Need procedure and function types for use in
RR-0096B 12.4.1 Allow a procedure body to be provided by a renaming declaration
RR-0286D 6.3 Interrupts should be handled with a procedure model, not a task model
AI-00605 4.6 the line, which is inconsistent with other GET procedures /skips terminators at the end of
RR-0316 6.3 Improve interrupt handling, e.g., with interrupt procedures
RR-0489 13.1.1 Allow machine-code insertions in functions as well as procedures
RR-0691 13.1.1 Allow machine-code insertions in functions as well as procedures
RR-0180 4.1 There is a need for procedures as parameters for X-Windows, etc
RR-0169 13.4 subprogram values Allow ''null'' procedures for actual or default generic formal
RR-0019 4.2 Allow types to specify finalization procedures for safely controlling use of/
RR-0665B 8.2 Support allocation of parallel processes to processors
RR-0071 13.2 Improve support for heterogeneous distributed processing
RR-0077 13.4 Provide stream I/O for digital signal processing
RR-0308 11.1 Add libraries for array processing
RR-0561 13.5.2 Allow case statement to operate on strings for string processing
RR-0421A 6.3 Need to delay in processing an interrupt
RR-0306 5.1 Need to be able to start processing at a particular time of day
RR-0738 7.3 Add facilities to support vector processing hardware
RR-0377 8.2 allow partitioning of programs for multiple processor environments Ada should
RR-0376 13.3 of exceptions in distributed/parallel/multi- processor systems Need special treatment
RR-0182 8.1 for parts of a program running on different processors Define visibility limits
RR-0665B 8.2 Support allocation of parallel processes to processors
RR-0372 13.2 Solve problem where heterogeneous processors view memory differently
RR-0626 6.2 portable among compilers, even for the/ Files produced by SEQUENTIAL_IO and DIRECT_IO are not
RR-0326 13.2 Use a different syntax production style
RR-0395 2.2 Include formal parameter names in parameter/result-type profile
RR-0600 2.2 Allow formal parameter names in parameter/result-type profile
RR-0229 13.4 to ensure these values are not used directly by programmers /and the initial value of an object
RR-0275 2.2 Error- prone and counter-intuitive aspects of RENAMES
RR-0581 12.2.1 the position of pragma ELABORATE are error- prone and unhelpful Rules specifying
RR-0529 13.5 of operations based on run-time queries about properties of types Allow selection
AI-00274 13.1 clause — record component visibility Proposed extension of the USE
RR-0063 5.3 performing critical functions Protect tasks from being aborted while
RR-0174 4.3 to be generic with respect to concurrency protection Allow packages
RR-0375 13.7 Include formal memory protection/security
RR-0295 4.6 for types other than string (make like PUT) Create TEXT_IO.PUT_LINE
AI-00873 2.3 Type conversion/ qualification of undefined scalar values
RR-0131 13.4 the enumeration literals of the/ In a qualified expression, should have visibility of
RR-0131 13.4 visibility of the enumeration literals of the qualifying type /expression, should have
RR-0314 13.7 Define minimum- quality error diagnostics in the standard
RR-0529 13.5 Allow selection of operations based on run-time queries about properties of types
RR-0328 9.3 Require compilers to report questionable uses of the language
RR-0075 5.2 Queue entries by task priority or FIFO based on application
RR-0655 5.4 Add asynchronous message queues
RR-0076 5.2 Allow selection of entry calls from entry queues and open alternatives based on/
RR-0415 5.2 Allow priority inheritance, prioritized entry- queues, and prioritized selective wait
RR-0072 5.2 real-time applications Prioritized queues and priority inheritance are needed for
RR-0657 5.2 Order entry queues based on priority
RR-0193 5.2 treatment of open select/ Allow priority queues, priority inheritance, and prioritized
RR-0015 5.2 Allow task priorities to control all queuing/select decisions
RR-0651 5.3 Allow one task to raise an exception in another task
RR-0200 12.3.4 Allow optional when_clause on RAISE and RETURN statements
RR-0135 13.4 Catenation should not raise CONSTRAINT_ERROR for intermediate results

RR-0751 12.3.4 Add WHEN/ RAISE construct to the language
RR-0362 12.3.4 Allow optional when_clause on the raise statement
RR-0132 12.3.4 Allow optional WHEN <condition> on RAISE statement for consistency with EXIT statement
RR-0141 12.3.4 Allow WHEN <condition> on RAISE statements
RR-0407A 4.5 Need exception name, line number, and unit name where raised
RR-0444 13.4 limit the places where a given exception can be raised Let the user
RR-0582 4.5 info about state when an exception is raised /additional implementation-dependent
RR-0621A 4.5 Need to find out which exception has been raised
RR-0468 4.4 No generic way to handle exceptions raised by generic formal subprograms
RR-0033A 4.5 Need to find the name of a raised exception
RR-0477 4.5 Provide a way to get the name and location of a raised exception
RR-0526C 4.5 Need to determine the name of a raised exception
RR-0219 4.5 Provide a way to get the name of the last raised exception, including an out-of-scope/
RR-0209 2.3 Require the compiler to report certain-to-be- raised exceptions
AI-00450 5.3 Should allow raising of an exception in another task
AI-00519 2.2 Default SMALL should be a power of two times the range
RR-0256 13.1 Fixed-point approach with range and delta is not what is needed
RR-0623 12.3.3 Define RANGE attribute for discrete ranges
RR-0155 12.3.3 Define RANGE attribute for scalar types
RR-0304 12.3.3 Define RANGE attribute for scalar types
AI-00584 13.4 Restrict argument of RANGE attribute in Ada 9x
RR-0252D 2.2 Fixed point type should include the bounds of the range definition
AI-00140 12.3.12 Allow -1..10 as a discrete range in loops
RR-0229 13.4 an object to ensure these/ Need to hide the range of a scalar type and the initial value of
RR-0571B 2.1 /when the choice in an aggregate is outside the range of the applicable index constraint
RR-0250 12.1.2 Define clearer notation for expressing null ranges
RR-0623 12.3.3 Define RANGE attribute for discrete ranges
RR-0046 13.5.3 Allow testing in discontiguous ranges and create true sets
RR-0249 12.1.2 ’First and ’last for null ranges are defined oddly
RR-0234 12.1.2 implementation burden ‘‘Sub-null’’ ranges are of little value and an
RR-0425 13.1 Need open ranges in declarations of real subtypes
RR-0357 10.1 Need packed decimal, wide- ranging fixed-point, decimal deltas
RR-0037 5.2 (i.e., delays) to execute using simulated time rather than a real-time clock Allow tasks
RR-0318 2.1 (with embedded mark-up) Make a machine- readable version of the Standard available
RR-0320 13.5 Generalize case statement for other types, including REAL
RR-0252B 11.1 whether rounding or truncation is used in real calculations /needs to know/control
AI-00262 2.2.8 and division Real literals with fixed point multiplication
RR-0127 13.4 Allow real number output in non-decimal bases
RR-0102 11.1 Provide explicit remainder operator for real numbers
RR-0348 11.1 Need predefined functions for real numbers, e.g., trig, log, etc
RR-0591 2.2.8 Allow fixed-point multiply/divide with universal real operands
RR-0425 13.1 Need open ranges in declarations of real subtypes
RR-0454 11.1 Need Entier function or attribute for real types
RR-0363 12.3.1 Allow ’VALUE and ’IMAGE to apply to real types as well as discrete types
RR-0317 2.2.12 Augment Ada’s looping: over reals, list items, etc
RR-0759 13.3 control engineering Add real-time and verification facilities for
RR-0072 5.2 Prioritized queues and priority inheritance are needed for real-time applications
RR-0037 5.2 to execute using simulated time rather than a real-time clock Allow tasks (i.e., delays)
RR-0493 4.2 should be able to ensure that storage will be reclaimed A programmer
RR-0451 4.3 Changes to package constants should not cause recompilation
RR-0368A 4.3 Ensure unnecessary recompilation is avoided
RR-0142 4.3 Reduce cases where recompilation of subunits is needed
RR-0688 4.3 subprogram body Unnecessary recompilation required when redeclaring a
RR-0723 8.2 Need support for reconfiguration in emergency cases
RR-0370A 8.2 Can’t recover space declared in library units when reconfiguring a system
RR-0596 12.3.13 Allow END type_name to substitute for END RECORD
RR-0673 12.3.13 Allow ‘‘END RECORD type_name’’ to substitute for ‘‘END’’ RECORD)
RR-0341 2.2 Allow discriminant value in record aggregate to be non-static
RR-0573 12.3.11 component initialization and as components of record aggregates /array aggregates for record
RR-0531 4.3 be usefully supported with current variant record approach Variants of a type can’t
AI-00539 2.2.4 Allow use of array/ record attributes in representation clauses
AI-00429 13.4.4 Allow array type definition for record component
RR-0573 12.3.11 Slide indices of array aggregates for record component initialization and as/
RR-0086 13.4 Need to initialize a record component to the address of the record itself
AI-00274 13.1 Proposed extension of the USE clause — record component visibility
RR-0321 13.4 Permit anonymous array and record declarations for record components
RR-0443 13.4.4 Need for anonymous array types as record components
RR-0749 12.3.11 serving as actual parameters and as values in record components /index sliding for slices

RR-0532 2.2.6 Allow same-type record components in different variants to share name
RR-0321 13.4 Permit anonymous array and record declarations for record components
RR-0212 13.6 Allow assignment to record discriminant like other components
RR-0522 2.2 Allow non-discrete record discriminants
RR-0593 4.6 Mandate implementation of variant record I/O in DIRECT_IO/SEQUENTIAL_IO
RR-0086 13.4 a record component to the address of the record itself Need to initialize
RR-0146 13.1 Support for file/ record locking
RR-0411 2.4 machine-independent way Express record representation clauses in a
RR-0290 13.5 The syntax used in record representation clauses is hard to read
RR-0289 6.2 Need multiple views of a record structure even when no discriminant is present
AI-00681 13.4 Can’t declare a constant of a NULL record type
AI-00345 4.6 Record type with variant having no discriminants
RR-0673 12.3.13 Allow ‘‘END’’ RECORD type_name to substitute for ‘‘END RECORD’’
RR-0505A 4.3 Provide extendable record types
RR-0568 2.2 Allow non-nested variant parts in record types
RR-0722 4.4 Need generic formal record types
AI-00452 4.4 Allow record types as generic formal parameters
RR-0248 13.1 locations for discriminants that are outside record values Allow users to specify
RR-0473 13.5 constrained subtypes of discriminated records Allow ‘‘partially’’
RR-0627 4.4 Allow partial match to formal type for records
RR-0649 2.2.2 Allow default initialization for all types (not just records)
RR-0053 13.4 Allow aggregates for null records and arrays
RR-0505B 4.4 Allow partial match for records as generic parameters
RR-0773 6.2 Need to pack variable-length records into a block for data transmission
RR-0336 13.4.4 Allow array type definitions in records; nice for array-of-array case
RR-0381 2.2 respect to components Records should have composed operations with
RR-0370A 8.2 reconfiguring a system Can’t recover space declared in library units when
RR-0370E 4.2 tasks are created by an allocator Need to recover space for task control blocks when
RR-0111 8.1 Provide explicit support for fault tolerance and recovery
RR-0118 4.2 storage reserve for STORAGE_ERROR recovery Provide a user-specified
RR-0490 2.3 Need successful/convenient recovery from exceptions in machine code insertions
RR-0269 13.6 Make subprograms not recursive by default
RR-0608 13.3 Allow recursive generic instantiations
RR-0533 4.3 be done Mutually recursive types from different packages cannot
RR-0688 4.3 Unnecessary recompilation required when redeclaring a subprogram body
RR-0424 13.6 Allow names exported from an instance to be redefined during instantiation
RR-0134 13.6 Require re-evaluation of entry’count on abandoned entries
RR-0251 13.6 notations to distinguish function call, array reference, and conversions Invent new
RR-0544 4.2 Need indivisible update on reference counts
RR-0524 6.4 of objects; allow programmer to ensure pass by reference for any object /to components
RR-0119 7.1 Need synchronized reference to elements of shared composite objects
RR-0206 2.1 Paragraph numbers should be included in the cross references
RR-0309 2.1 Ensure all cross references are complete and correct
RR-0524 6.4 programmer to ensure/ Allow functions to return references to components of objects; allow
RR-0194 12.1.1 Disallow referencing a task from outside its master
RR-0030 13.6 specification to list non-local objects referred to Require subprogram
RR-0720 11.1 Floating-point model should reflect actual hardware architectures
RR-0252E 11.1 Provide a floating point model that reflects actual machine architecture
RR-0718 9.1 results in numeric computation, especially regarding optimization Need predictable
RR-0110 6.4 of and access to data in different types or regions of memory /control over placement
RR-0510 2.2.10 Re-indexing arrays via type conversions
RR-0122 2.2 Permit an implementation to reject some integer types as array indexes
AI-00850 2.3 are not met Rejecting a unit when a pragma’s assumptions
RR-0020 5.2 during program execution, so priorities should/ Relative importance of functions may change
RR-0466 4.2 finalization for objects of a type to ensure release of resources Allow user-defined
RR-0523 4.2 finalization for objects of a type to ensure release of resources Allow user-defined
RR-0676 4.2 Add finalization to ensure release of resources
AI-00570 12.1.1 type instances Releasing heap storage associated with task
RR-0737 5.2 alternatives in a select statement Allow reliable user control over selection of
RR-0102 11.1 Provide explicit remainder operator for real numbers
RR-0467 12.2.3 Need convenient way to rename a type and get its operations
RR-0382 4.6 Need to be able to rename and append to a file in standard Ada
RR-0231 12.4.1 Allow a rename definition of a subprogram body
RR-0725 12.4.1 Need rename in package body for routine in package spec
RR-0774M 13.1 Allow a subprogram to be renamed in a body
RR-0239B 2.2.3 parameter type conversion A renamed type cannot be used in an actual
RR-0275 2.2 Error-prone and counter-intuitive aspects of RENAMES
RR-0601 2.2 Allow library-level declarations to be defined by RENAMES

RR-0667 12.4.1 Allow a subprogram body to be given by RENAMES
RR-0764 12.4.1 Allow subprogram bodies to be defined by RENAMES
RR-0610 2.2 Why not allow RENAMES for types and subtypes?
RR-0550 12.4.1 Allow subprogram bodies to be defined by RENAMES or generic instantiation
RR-0393 12.2.3 of fixed point mult and div operator by renaming Can’t get direct visibility
RR-0096A 12.2.3 Permit renaming an enumeration literal as a character literal
RR-0239A 12.2.3 literals visible Renaming an enumeration type should make
RR-0096B 12.4.1 Allow a procedure body to be provided by a renaming declaration
RR-0096C 12.4.2 of a private type to be provided by a renaming declaration /the full declaration
RR-0557 4.3 bodies helps get around the/ The use of renaming declarations to provide subprogram
RR-0570 13.1 Allow the prefix of a name to denote a renaming of an enclosing construct
RR-0055 12.4.1 Allow a subprogram body to be defined by renaming or generic instantiation
RR-0470 12.4.1 subprogram body Allow renaming or generic instantiation to define a
RR-0157 12.4.1 Allow renaming when defining a subprogram body
RR-0185 5.2 General Ada rendezvous is slow; semaphores would be better
RR-0173 13.3 set of tasks Allow a rendezvous with a higher-level entity, i.e., a
RR-0065 4.3 from the/ To improve reuse possibilities, allow rep clauses and various pragmas to be separated
RR-0171 4.3 Allow target-dependent code (including rep clauses) to be separate from other code
AI-00216 10.2 case, control, etc., independent of character representation /are numeric, upper case, lower
RR-0252C 11.1 can choose appropriate floating point representation Ensure programmer
RR-0684 4.3 Related packages need access to a private type’s representation
RR-0048 2.2.4 Extend static expressions to include representation attributes of composite types
RR-0288 13.5 Integrate representation clause information with declarations
AI-00539 2.2.4 Allow use of array/record attributes in representation clauses
RR-0418 2.2 be added Representation clauses for array types need to
RR-0411 2.4 Express record representation clauses in a machine-independent way
RR-0565 2.2 ’SMALL is unsuitably defined; need for representation clauses inappropriate
RR-0290 13.5 The syntax used in record representation clauses is hard to read
RR-0007 2.4 Default representation for enumeration types should be specified
RR-0465 2.2.14 vice versa Need a way to get the representation from an enumeration value and
RR-0560 4.3 Need to access a private type’s representation in related packages
RR-0315 2.4 Allow integer type names that indicate representation size, e.g., INTEGER_32, to improve/
RR-0187 2.4 Need to allow unsigned enumeration representation specifications
RR-0225 11.1 Ensure floating point representation with desired accuracy is used
RR-0166 13.3 Allow definition of the literal representations of an abstract data type
RR-0059 2.2.14 Need an attribute for returning a representation’s underlying value
RR-0515 4.2 objects, especially in/ Need ability to request indivisible update for specific
RR-0150 13.7 of different programs to reduce memory requirements Provide ‘‘chaining’’
RR-0401 2.2 cannot be done efficiently because of accuracy requirements Mixed-base fixed-point operations
RR-0374 4.2 Ada should address memory management requirements in distributed systems
RR-0322 13.1 Do not add any new reserved words to the language
AI-00223 5.1 Require adequate resolution for the function CLOCK
RR-0724 2.1 Need clearer/simpler overload resolution rules, especially for implicit conversion
AI-00529 13.1 Resolving the meaning of an attribute name
RR-0466 4.2 for objects of a type to ensure release of resources Allow user-defined finalization
RR-0523 4.2 for objects of a type to ensure release of resources Allow user-defined finalization
RR-0676 4.2 Add finalization to ensure release of resources
RR-0478 13.1 Add language facilities for restricting use of resources to trusted packages
RR-0370B 8.2 Can’t restart library level tasks
AI-00584 13.4 Restrict argument of RANGE attribute in Ada 9x
RR-0478 13.1 Add language facilities for restricting use of resources to trusted packages
RR-0423 2.2 Remove discriminant restriction on full declarations of private types
RR-0427 12.1.1 Do not permit a function to return a locally-declared task object
RR-0352 5.1 Require Calendar.Clock to return consistently accurate local system time
RR-0524 6.4 allow programmer to ensure/ Allow functions to return references to components of objects;
RR-0620 13.6 Ban RETURN statement except inside functions
RR-0200 12.3.4 Allow optional when_clause on RAISE and RETURN statements
RR-0614 12.3.4 Allow WHEN condition RETURN to make selection of returned value clearer
AI-00487 4.6 to be/ END_OF_PAGE and END_OF_FILE should not return TRUE when there is still an empty line
RR-0614 12.3.4 Allow WHEN condition RETURN to make selection of returned value clearer
RR-0059 2.2.14 Need an attribute for returning a representation’s underlying value
RR-0513 12.3.9 Allow overloading of = for any type, e.g., returning an array type
RR-0255 11.1 Provide a function for returning the value of the next floating point number
RR-0383 4.4 Need generic exceptions for truly reusable generic units
RR-0065 4.3 various pragmas to be separated/ To improve reuse possibilities, allow rep clauses and
RR-0066 2.3 execution/incorrect order dependences Reduce risks associated with erroneous
RR-0014 4.1 Need to call subprograms loaded in ROM
RR-0136 6.1 Provide support for bit-field operations such as shift, rotate

RR-0139 6.1 Provide shift and rotate operations for boolean arrays
RR-0252B 11.1 Programmer needs to know/control whether rounding or truncation is used in real/
AI-00526 2.4 Rounding up or down
RR-0409 2.4 Define in the language how 3.5 rounds to integer
RR-0213 2.4 Need to be able to find out if an implementation rounds up or down
RR-0725 12.4.1 Need rename in package body for routine in package specification
RR-0669 4.2 Allow user-written := routines
AI-00488 4.6 Skipping of leading line terminators in GET routines causes problems in interactive I/O
RR-0475 4.2 Need automatically-invoked user-defined routines to reclaim storage
RR-0507 11.2 Provide information/control over row-major or column-major ordering
RR-0637 11.1 Ada programs should run as though negative zero did not exist
RR-0243 8.2 Allow/require elaboration prior to run time
RR-0450 6.4 of buffers whose type is determined at run time Need efficient manipulation
RR-0246 8.2 /that constant declarations are not elaborated at run time when initialized with static/
RR-0735 6.3 Need ability to change interrupt bindings at run-time
RR-0285 8.2 Minimize the need for run-time elaboration
RR-0497 13.7 used as generic actual can yield a surprising run-time error /default discriminants for types
RR-0242 2.3 Require compilation warnings for potential run-time errors
RR-0244B 2.3 Flag run-time errors at compile-time when possible
RR-0529 13.5 Allow selection of operations based on run-time queries about properties of types
RR-0074 5.2 Define a standard run-time support environment interface
RR-0286B 5.2 access to interrupts that are also used by the run-time system Embedded system user may need
RR-0421D 6.3 calls may depend inappropriately on the run-time system /timed, or conditional
RR-0279 2.2 include code for/ If tasks are not used, the run-time system and compiled code should not
RR-0175 5.2 Define interface between compiler- and target-specific run-time system aspects
RR-0728 8.1 Need simple Ada run-time system for distributed memory MIMD architecture
RR-0176 9.1 allocation strategies Document run-time system performance and memory
RR-0286C 5.2 privileged mode Run-time system should avoid entering
RR-0338 6.4 Provide pointers to static objects and safe conversion between ADDRESS values and access/
RR-0564 11.1 include more mantissa digits in floating point safe numbers Allow implementation freedom to
AI-00812 2.2 Attributes SAFE_LARGE and SAFE_SMALL should be static
RR-0019 4.2 Allow types to specify finalization procedures for safely controlling use of collections
AI-00812 2.2 Attributes SAFE_LARGE and SAFE_SMALL should be static
RR-0435 9.3 Need secondary standard for simple Ada subset for safety-critical applications
RR-0229 13.4 to ensure these/ Need to hide the range of a scalar type and the initial value of an object
RR-0677 2.2.2 Allow initialization clauses on scalar type declarations
RR-0155 12.3.3 Define RANGE attribute for scalar types
RR-0304 12.3.3 Define RANGE attribute for scalar types
AI-00873 2.3 Type conversion/qualification of undefined scalar values
RR-0693 2.2 Parameter passing rules for scalars makes generic code sharing hard
RR-0656 5.1 Need timed exceptions for deadline scheduling
RR-0379 5.2 Application should select the specific scheduling algorithm
RR-0016 5.2 Allow user-selectable task scheduling algorithms
RR-0170 5.2 Permit or provide alternate scheduling algorithms
RR-0124 5.2 Ensure that code dependent on task scheduling algorithms is portable
RR-0121 5.2 Provide more user control over scheduling decisions
RR-0278 5.2 Tasking model should support common scheduling disciplines more easily
RR-0219 4.5 the last raised exception, including an out-of- scope exception /a way to get the name of
RR-0740 2.2 to inlined subprograms, allow merging of scopes For optimization with respect
RR-0763 2.3 Allow nested scopes to turn off pragma SUPPRESS
RR-0089 4.6 Provide facilities for I/O screen operations
RR-0351 13.7 Trusted systems require auto- scrubbing of memory when done with it
RR-0435 9.3 safety-critical applications Need secondary standard for simple Ada subset for security
RR-0375 13.7 Include formal memory protection/
RR-0647 4.1 case statements Need ability to select actions depending on state without using
RR-0193 5.2 inheritance, and prioritized treatment of open select alternatives /priority queues, priority
RR-0199 12.3.13 Allow IF, CASE, and SELECT constructs to be named
RR-0015 5.2 Allow task priorities to control all queuing/ select decisions
RR-0737 5.2 control over selection of alternatives in a select statement Allow reliable user
RR-0340 12.3.13 Allow optional simple name on CASE, IF, and SELECT statements
RR-0379 5.2 Application should select the specific scheduling algorithm
RR-0016 5.2 Allow user- selectable task scheduling algorithms
RR-0462 12.3.7 formal part even when the selected/ Allow selected component form of type mark in a
RR-0462 12.3.7 /of type mark in a formal part even when the selected component has the same identifier as/
RR-0737 5.2 Allow reliable user control over selection of alternatives in a select statement
RR-0076 5.2 open alternatives based on priorities Allow selection of entry calls from entry queues and
RR-0529 13.5 queries about properties of types Allow selection of operations based on run-time
RR-0614 12.3.4 Allow WHEN condition RETURN to make selection of returned value clearer

RR-0679 13.4 Allow component selection on objects of a private type
RR-0575 2.2.9 Need better (more selective) control over inlining
RR-0624 12.2.3 Provide selective direct visibility into a package
RR-0727 12.2.3 Need selective direct visibility of package declarations
RR-0398 2.2.9 Need clearer/more selective rules for pragma INLINE applicability
RR-0555 12.2.3 and subprograms of a type Need ‘‘selective’’ USE clause to get just operators
RR-0415 5.2 prioritized entry-queues, and prioritized selective wait Allow priority inheritance,
RR-0612 13.3 Should allow both delay and terminate alternatives in selective wait
RR-0697 5.2 Allow entry call alternative in selective wait
RR-0083 5.3 transfer of control via entry call/ selective wait construct Provide asynchronous
RR-0498 5.2 accept statements and entry calls Make selective wait symmetrical with respect to
RR-0292 2.1 Section 13.6 of the standard has no semantic content
RR-0732 2.4 an integer type Clarify semantics of instantiating ENUMERATION_IO with
RR-0109 8.1 single distributed Ada program Provide Ada semantics that are helpful when dealing with a
RR-0461 5.2 Provide standard package of semaphore operations
RR-0185 5.2 General Ada rendezvous is slow; semaphores would be better
RR-0028 13.4 Add a semicolon terminator to SEPARATE statement syntax
RR-0392 13.5 predefined = Need ‘‘semi-limited’’ type with predefined := but no
AI-00003 13.1 Allow data of mode IN in SEND_CONTROL
RR-0480 8.1 Need standard means of sending messages between Ada programs
RR-0237 4.3 particular library model Make separate compilation independent of a
RR-0562 4.4 and bodies Require separate compilation of generic specifications
RR-0171 4.3 /target-dependent code (including rep clauses) to be separate from other code
RR-0537 13.6 in Pascal Separate integer divide and floating divide as
RR-0698 4.3 separate units Need ability to separate portable and non-portable code into
RR-0774I 13.1 Create separate standards, such as X-Windows, SQL
RR-0028 13.4 Add a semicolon terminator to SEPARATE statement syntax
RR-0698 4.3 to separate portable and non-portable code into separate units Need ability
RR-0065 4.3 /allow rep clauses and various pragmas to be separated from the compilation unit to which/
RR-0208 13.4 Need ability to initiate TEXT_IO, DIRECT_IO, and SEQ_IO operations without waiting for/
RR-0520 13.1 Language should distinguish ‘‘sequence’’ and ‘‘mapping’’ arrays
RR-0593 4.6 of variant record I/O in DIRECT_IO/ SEQUENTIAL_IO Mandate implementation
RR-0626 6.2 among compilers, even for/ Files produced by SEQUENTIAL_IO and DIRECT_IO are not portable
RR-0021 5.2 Need priority inheritance for server tasks
RR-0347 5.2 priority to increase as a function of lack of service /under program control; allow task
RR-0749 12.3.11 Should allow index sliding for slices serving as actual parameters and as values in record/
RR-0031 13.5.3 Provide a way to test for a value in a non-contiguous set
RR-0034 3.1 Ada should use ISO 8859/1-9 (8-bit) character set
RR-0148 3.1 for extended and graphic characters (256 ASCII set) Provide support
RR-0438 3.1 Allow use of multi-octet character set
RR-0311 3.1 Generalize character set for 8-bit characters
RR-0283 4.3 Need convenient way to set global compilation parameters
RR-0173 13.3 rendezvous with a higher-level entity, i.e., a set of tasks Allow a
RR-0370D 5.2 Need to set priorities of tasks during mode shifts
RR-0464 12.3.5 Should be able to set STORAGE_SIZE for task objects as well as types
RR-0648 12.3.5 Need to set STORAGE_SIZE on task objects, not task types
RR-0105 5.1 Allow application to set/adjust clocks
RR-0046 13.5.3 Allow testing in discontiguous ranges and create true sets
RR-0367 3.1 Need support for national language character sets, including string comparison
RR-0448 4.3 Allow different sets of subprograms to depend on common declarations
RR-0481 2.1 Make Ada documentation available in SGML format
RR-0532 2.2.6 record components in different variants to share name Allow same-type
RR-0678 7.1 Pragma SHARED is not sufficient for data shared between programs; need VOLATILE
RR-0119 7.1 Need synchronized reference to elements of shared composite objects
RR-0678 7.1 between programs; need VOLATILE Pragma SHARED is not sufficient for data shared
RR-0521 5.2 Need more convenient support for use of shared memory among tasks
AI-00142 7.1 Allow pragma SHARED to be applied to components of composite objects
RR-0434 7.1 Need atomic read/write operations on shared volatile memory
RR-0342 4.4 Do not implement requests that will break generic code sharing
RR-0693 2.2 Parameter passing rules for scalars makes generic code sharing hard
RR-0005 4.4 Exception declarations in generic packages make code sharing unnecessarily difficult
RR-0139 6.1 Provide shift and rotate operations for boolean arrays
RR-0766 6.1 Allow bit-wise operations (AND, SHIFT) on integers, bytes, etc
RR-0634 6.1 Provide arithmetic shift operations for integers
RR-0136 6.1 Provide support for bit-field operations such as shift, rotate
RR-0370D 5.2 Need to set priorities of tasks during mode shifts
RR-0280 5.1 unnecessary; timing performance must be/ Short delays are too inefficient; Calendar time
RR-0265 13.7 Allow implementations to short-circuit in general, forget AND THEN

RR-0061 2.4 Make Long_Float and Short_Float required types
RR-0517 9.3 Provide syntax to declare program units free from side-effects
RR-0453 11.1 Provide a special function or attribute yielding the sign of a numeric value
RR-0077 13.4 Provide stream I/O for digital signal processing
RR-0120 4.2 Allow users to defer the signalling of STORAGE_ERROR when space is exhausted
RR-0037 5.2 Allow tasks (i.e., delays) to execute using simulated time rather than a real-time clock
RR-0700 13.1 Ensure that constant functions like sin(10.0) are evaluated at compile-time
RR-0060 2.2.9 of subprograms from some but not all call sites Allow inlining
RR-0774E 4.5 Provide access to context of an exception situation
RR-0315 2.4 /integer type names that indicate representation size, e.g., INTEGER_32, to improve portability
RR-0464 12.3.5 Should be able to set STORAGE_ SIZE for task objects as well as types
AI-00453 12.3.5 STORAGE_ SIZE for tasks
RR-0717 2.2.12 Allow specification of a step size in FOR loops
RR-0463 13.4 ’ Size is unclear; perhaps need ’Spacing and ’Allocation
RR-0648 12.3.5 Need to set STORAGE_ SIZE on task objects, not task types
RR-0703 12.3.5 Need to specify STORAGE_ SIZE on task objects, not task types
RR-0018 6.4 Need pre-elaborated constant arrays with variable-sized elements
RR-0138 6.1 Need full-sized unsigned integers
RR-0553 4.6 GET_LINE should not automatically call SKIP_LINE
AI-00488 4.6 routines causes problems in interactive I/O Skipping of leading line terminators in GET
AI-00605 4.6 is inconsistent with other GET/ GET_LINE skips terminators at the end of the line, which
RR-0323 13.4.2 Generalize slice for multidimensional arrays
RR-0494 13.4.2 Allow slices for any dimension in multidimensional arrays
RR-0508 13.4.2 Allow slices for any dimension in multidimensional arrays
RR-0240 12.3.11 Non-sliding aggregates and slices in component associations
RR-0749 12.3.11 values in/ Should allow index sliding for slices serving as actual parameters and as
RR-0573 12.3.11 component initialization and as components of/ Slide indices of array aggregates for record
RR-0240 12.3.11 Non- sliding aggregates and slices in component associations
RR-0749 12.3.11 and as values in record/ Should allow index sliding for slices serving as actual parameters
AI-00521 13.3 Fixed point subtypes should not inherit SMALL
RR-0565 2.2 representation clauses inappropriate ’ SMALL is unsuitably defined; need for
AI-00519 2.2 Default SMALL should be a power of two times the range
AI-00812 2.2 Attributes SAFE_LARGE and SAFE_ SMALL should be static
RR-0210 13.7 Need more pragmas for software maintenance to MIL standards
RR-0087 6.3 Allow software priorities to match/exceed hardware priorities
RR-0162 13.1 Provide a clean interface to a SORT package
RR-0339 13.1 Support sorting in extended alphabets
RR-0746 13.7 Allow pictures/graphics as comments in source code
RR-0370A 8.2 reconfiguring a system Can’t recover space declared in library units when
RR-0370E 4.2 created by an allocator Need to recover space for task control blocks when tasks are
RR-0495 13.6 Remove leading space in the result of the ’IMAGE attribute for integers
RR-0120 4.2 to defer the signalling of STORAGE_ERROR when space is exhausted Allow users
RR-0112 4.2 Provide user support for controlled space reclamation
RR-0463 13.4 ’Size is unclear; perhaps need ’ Spacing and ’Allocation
RR-0082 2.2.5 of objects of private types in visible package specification Allow declaration
RR-0675 12.3.7 identifier to be used as a type mark in its specification Allow a subprogram
RR-0725 12.4.1 Need rename in package body for routine in package specification
RR-0268 13.6 Separation of specification and body is not worth it
RR-0153 13.1 Private part foils separation of specification and implementation
RR-0717 2.2.12 Allow specification of a step size in FOR loops
RR-0471 13.6 calls for clarity Allow specification of parameter modes in subprogram
RR-0270 13.4 Allow specification of read-only data from a package
RR-0701 13.4 for SYSTEM Allow specification of STANDARD in the same way as
RR-0030 13.6 referred to Require subprogram specification to list non-local objects
RR-0547 2.2 Like non-generic subprograms, allow merge of specification/body for generic subprograms
RR-0604 2.2 Like non-generic subprograms, allow merge of specification/body for generic subprograms
AI-00280 2.2 Allow pragma OPTIMIZE in package specifications
RR-0187 2.4 Need to allow unsigned enumeration representation specifications
RR-0562 4.4 Require separate compilation of generic specifications and bodies
RR-0267 2.1 The Standard is confusing in distinguishing specifications and declarations
RR-0581 12.2.1 error-prone and unhelpful Rules specifying the position of pragma ELABORATE are
RR-0774I 13.1 Create separate standards, such as X-Windows, SQL
RR-0719 11.1 Need standard for trig functions, sqrt, etc
AI-00510 3.1 Use ISO symbols and standards in the Ada ISO Standard
RR-0091 4.3 Don’t specify the compilation process in the Standard
RR-0299 13.1 Make everything in the Standard ‘‘part of the’’ standard
RR-0314 13.7 Define minimum-quality error diagnostics in the standard
RR-0382 4.6 Need to be able to rename and append to a file in standard Ada

RR-0252A 11.1 Ensure support for IEEE floating point standard; allow full use of machine/
RR-0731 11.1 Use the Language Compatible Arithmetic Standard as a basis for Ada's floating point model
RR-0318 2.1 Make a machine-readable version of the Standard available (with embedded mark-up)
RR-0084 5.2 permit high-performance/ Specify standard conventions for using tasks that
RR-0435 9.3 safety-critical applications Need secondary standard for simple Ada subset for
RR-0719 11.1 Need standard for trig functions, sqrt, etc
RR-0292 2.1 Section 13.6 of the standard has no semantic content
RR-0369 11.1 Provide support for floating point standard IEEE-754
RR-0701 13.4 Allow specification of STANDARD in the same way as for SYSTEM
AI-00485 4.6 for interactive I/O Having independent standard input and output files is not useful
RR-0582 4.5 implementation-dependent info about/ Provide standard interface for getting additional
RR-0267 2.1 specifications and declarations The Standard is confusing in distinguishing
RR-0683 2.2 Section 11.6 of the Standard is unclear about what replacements are allowed
RR-0260 2.1 The Standard is unclear in various ways
RR-0181 8.1 Need standard means of communicating between Ada programs
RR-0378 8.1 Need standard means of communication in distributed system
RR-0480 8.1 Need standard means of sending messages between programs
AI-00216 10.2 are numeric, upper case, lower case,/ Provide standard methods for testing whether characters
AI-00582 13.4 Need a standard name for null address
RR-0748 5.4 Provide standard package of asynchronous primitives
RR-0774B 13.6 Tasking defined as a standard package of functions
RR-0159 4.6 Add standard package of general file system functions
RR-0461 5.2 Provide standard package of semaphore operations
RR-0299 13.1 Make everything in the Standard ''part of the standard''
RR-0074 5.2 Define a standard run-time support environment interface
RR-0501 2.1 section headings The Standard should be consistent in delimiting
RR-0502 2.1 upper and lower cases The Standard should be consistent in its use of
RR-0068 2.4 support is optional for embedded systems The Standard should explicitly acknowledge that I/O
RR-0189 11.1 library interface Standard should include a floating-point math
RR-0644 9.1 for certain operations Standard should specify time bounds/constraints
RR-0622 2.1 generic formal types The Standard should use ''metatype'' in describing
RR-0051B 10.4 Provide standard string manipulation packages
RR-0479 13.1 information from OS Need standard subprograms to get user-interface
RR-0590 5.2 Need clear, efficient, standard support for mutual exclusion
RR-0151 6.3 Need standard support for priority interrupts
RR-0245 8.2 Change Standard to encourage pre-elaboration
RR-0386 9.1 Need standard way of telling the compiler not to optimize
RR-0137 2.4 Standardize bit storage/order conventions
RR-0527 4.1 pragma INTERFACE Standardize information/conventions used for
RR-0177 4.3 library for configuration management Standardize interface between compiler and
RR-0355 2.4 line arguments Standardize means of getting the OS command
RR-0681 13.7 A definition of an Ada Line Of Code (LOC) should be standardized
RR-0345 13.1 Need standardized interface to other ANSI languages
RR-0602 met Encourage implementors to support standardized libraries
RR-0226 4.3 management capabilities Need standardized support for improved library
RR-0210 13.7 Need more pragmas for software maintenance to MIL standards
RR-0325A 9.3 Allow implementations to enforce local coding standards
AI-00510 3.1 Use ISO symbols and standards in the Ada ISO Standard
RR-0774I 13.1 Create separate standards, such as X-Windows, SQL
RR-0306 5.1 Need to be able to start processing at a particular time of day
RR-0132 12.3.4 on RAISE statement for consistency with EXIT statement Allow optional WHEN <condition>
RR-0216 9.3 Require that each task entry have at least one accept statement
RR-0281 2.1 Confusing treatment of term ''delay'' statement
RR-0362 12.3.4 Allow optional when_clause on the raise statement
RR-0491 13.1.2 Code would be clearer if one could EXIT from a block statement
RR-0538 9.3 Create new loop structure which bans the EXIT statement
RR-0618 13.1 Ban GOTO statement
RR-0737 5.2 over selection of alternatives in a select statement Allow reliable user control
RR-0650 13.5.2 Allow non-static case statement choices, non-discrete case statement expression
RR-0620 13.6 Ban RETURN statement except inside functions
RR-0650 13.5.2 case statement choices, non-discrete case statement expression Allow non-static
RR-0632 13.1.2 Allow EXIT from a block statement for consistency
RR-0132 12.3.4 Allow optional WHEN <condition> on RAISE statement for consistency with EXIT statement
RR-0320 13.5 Generalize case statement for other types, including REAL
RR-0335 5.3 Effect of abort statement is too implementation-dependent
RR-0658 5.2 Allow accept statement possibility in a conditional entry call
RR-0028 13.4 Add a semicolon terminator to SEPARATE statement syntax
RR-0312 13.5 Generalize case statement to decision table

AI-00211	13.4	Additional control	statement to hop to end of the loop
RR-0561	13.5.2	Allow case	statement to operate on strings for string processing
RR-0141	12.3.4	Allow WHEN <condition> on RAISE	statements
RR-0200	12.3.4	Allow optional when_clause on RAISE and RETURN	statements
RR-0340	12.3.13	Allow optional simple name on CASE, IF, and SELECT	statements
RR-0499	12.3.2	‘‘blocks’’, allow exception handlers in accept	statements Like other
RR-0647	4.1	actions depending on state without using case	statements Need ability to select
RR-0498	5.2	wait symmetrical with respect to accept	statements and entry calls Make selective
AI-00214	2.2.7	Allow accept	statements in program units nested in tasks
RR-0543	2.2.7	Allow accept	statements in subprograms nested inside tasks
RR-0621C	13.4	Allow case	statements to dispatch on value of an exception
AI-00477	13.5.2	Case choices should not have to be	static
AI-00812	2.2	Attributes SAFE_LARGE and SAFE_SMALL should be	static
RR-0271	13.6	for variables with key words like CONTROLLED or	STATIC Distinguish storage classes
RR-0341	2.2	value in record aggregate to be non-	static Allow discriminant
RR-0650	13.5.2	case statement expression Allow non-	static case statement choices, non-discrete
RR-0009	12.3.6	Allow static conversion to	static discrete type of static discrete expression
RR-0099	12.3.6	Explicit type conversions should be allowed in	static expressions
RR-0246	8.2	elaborated at run time when initialized with	static expressions /declarations are not
RR-0705	2.2	For better performance, remove restrictions on	static expressions
RR-0452	13.4	Allow constant functions in	static expressions (or overloadable constants)
RR-0048	2.2.4	attributes of composite types Extend	static expressions to include representation
RR-0227	4.4	Allow generic parameterization with	static numeric quantities
RR-0338	6.4	ADDRESS values and access/ Provide pointers to	static objects and safe conversion between
RR-0726	6.4	Need non-contiguous arrays,	static pointers
RR-0654	5.2	Need non-	static priorities
RR-0616	2.3	Require compilers to diagnose	statically-detectable constraint errors
RR-0445	4.4	Non-	staticness of generic formals poses problems
RR-0717	2.2.12	Allow specification of a	step size in FOR loops
RR-0742	5.3	Need ability to asynchronously	stop another task
RR-0431	5.3	A terminate alternative cannot be used to	stop cyclic tasks
RR-0768	5.3	Need to asynchronously interrupt another task to	stop it
RR-0475	4.2	user-defined routines to reclaim	storage Need automatically-invoked
AI-00570	12.1.1	Releasing heap	storage associated with task type instances
RR-0271	13.6	like CONTROLLED or STATIC Distinguish	storage classes for variables with key words
RR-0113	4.2	Ensure that there are no	storage ‘‘leaks’’
RR-0168	4.2	Allow implicitly-invoked finalization code for	storage management
RR-0702	4.2	There is a need for improvements in heap	storage management
RR-0118	4.2	Provide a user-specified	storage reserve for STORAGE_ERROR recovery
RR-0291	6.4	Clarify whether use of an address clause causes	storage to be initialized
RR-0017	6.2	Be able to treat an Ada object as an array of	storage units
RR-0493	4.2	A programmer should be able to ensure that	storage will be reclaimed
RR-0118	4.2	Provide a user-specified storage reserve for	STORAGE_ERROR recovery
RR-0120	4.2	Allow users to defer the signalling of	STORAGE_ERROR when space is exhausted
RR-0137	2.4	Standardize bit	storage/order conventions
RR-0464	12.3.5	Should be able to set	STORAGE_SIZE for task objects as well as types
AI-00453	12.3.5		STORAGE_SIZE for tasks
RR-0648	12.3.5	Need to set	STORAGE_SIZE on task objects, not task types
RR-0703	12.3.5	Need to specify	STORAGE_SIZE on task objects, not task types
RR-0176	9.1	system performance and memory allocation	strategies Document run-time
RR-0077	13.4	Provide	stream I/O for digital signal processing
RR-0310	10.4	Need convenient way to pad with blanks in	string assignments
RR-0367	3.1	for national language character sets, including	string comparison Need support
RR-0051C	10.4	Provide packages for	string edit functions
RR-0295	4.6	Create TEXT_IO.PUT_LINE for types other than	string (make like PUT)
RR-0324	10.4	Add more flexible support for	string manipulation
RR-0051B	10.4	Provide standard	string manipulation packages
RR-0561	13.5.2	Allow case statement to operate on	strings for string processing
RR-0054	13.2	Do not add variable length	strings to the language
RR-0327	10.4	Add varying length	strings to the language
RR-0419	10.4	Add some form of support for varying length	strings to the language
RR-0163	10.4	Need support for variable-length	strings with appropriate equality and assignment/
RR-0421B	6.3	/structure is sometimes different from memory address	structure; a single type for both is/
RR-0615	2.2.12	Define LOOP/UNTIL control	structure as in Pascal
RR-0289	6.2	Need multiple views of a record	structure even when no discriminant is present
RR-0282	13.3	Ada program	structure hides important context information
RR-0421B	6.3	address structure; a single/ Interrupt address	structure is sometimes different from memory
RR-0457	4.3	visibility of library units	Structure library units as groups, control

RR-0538 9.3 Create new loop structure which bans the EXIT statement
RR-0639 8.2 Need compile-time initialization of complex data structures
RR-0103B 6.2 Provide efficient means of reading large data structures in chunks
RR-0262 13.7 Do not require existence of subunit for body stubs
RR-0234 12.1.2 implementation burden ‘‘Sub-null’’ ranges are of little value and an
RR-0462 12.3.7 component has the same identifier as the subprogram /formal part even when the selected
RR-0081 4.1 Provide subprogram and package types
AI-00382 2.2 Allow generic subprogram bodies
RR-0557 4.3 The use of renaming declarations to provide subprogram bodies helps get around the/
RR-0764 12.4.1 Allow subprogram bodies to be defined by RENAMES
RR-0550 12.4.1 generic instantiation Allow subprogram bodies to be defined by RENAMES or
RR-0157 12.4.1 Allow renaming when defining a subprogram body
RR-0231 12.4.1 Allow a rename definition of a subprogram body
RR-0470 12.4.1 Allow renaming or generic instantiation to define a subprogram body
RR-0688 4.3 Unnecessary recompilation required when redeclaring a subprogram body
RR-0364 12.4.1 Allow a subprogram body to be defined by generic instantiation
RR-0055 12.4.1 generic instantiation Allow a subprogram body to be defined by renaming or
RR-0666 12.4.1 Allow a subprogram body to be given by generic instantiation
RR-0667 12.4.1 Allow a subprogram body to be given by RENAMES
RR-0388 4.1 Proposal for clean way of executing a subprogram by its address
RR-0458 4.4 Need convenient way to escape into weakly typed subprogram call
RR-0064 4.1 Allow some form of subprogram callback
RR-0629 4.1 Need procedure and function types for use in subprogram calls
RR-0471 13.6 Allow specification of parameter modes in subprogram calls for clarity
RR-0675 12.3.7 in its specification Allow a subprogram identifier to be used as a type mark
RR-0579 12.3.7 Allow a type mark of form P.FOO in the formal part of a subprogram named FOO
RR-0606 13.5.4 Allow generic subprogram names to be overloaded
RR-0414 4.1 Ada needs subprogram types and subprogram objects
RR-0214 13.1 Require that a subprogram parameter be used within the body
RR-0518 13.1 Provide syntax to declare subprogram pre/post conditions
RR-0030 13.6 objects referred to Require subprogram specification to list non-local
RR-0774M 13.1 Allow a subprogram to be renamed in a body
RR-0483 12.3.7 generic unit (as is/ Allow an instantiated subprogram to have the same identifier as the
RR-0430A 4.1 Need objects of a subprogram ‘‘type’’
RR-0414 4.1 Ada needs subprogram types and subprogram objects
RR-0563 4.1 Need to allow subprogram types and variables
RR-0503 4.1 Provide subprogram types for dispatcher-style programming
RR-0611 4.1 parameters, etc Allow subprogram types, variables, constants,
RR-0169 13.4 procedures for actual or default generic formal subprogram values Allow ‘‘null’’
RR-0384 5.1 specified delay Cannot write subprogram which causes an exception after
RR-0032 12.2.2 Allow grouping of variable declarations and related subprograms
RR-0101B 4.4 exceptions as parameters to generic units and subprograms Need to pass
RR-0426B 2.2 Allow declaration and body to be combined for generic subprograms
RR-0468 4.4 to handle exceptions raised by generic formal subprograms No generic way
RR-0488 4.4 formal entries as well as generic formal subprograms Allow generic
RR-0512 4.1 Provide subprograms as parameters to subprograms
RR-0526B 4.4 exceptions as parameters to generic units and subprograms Need to pass
RR-0547 2.2 allow merge of specification/body for generic subprograms Like non-generic subprograms,
RR-0604 2.2 allow merge of specification/body for generic subprograms Like non-generic subprograms,
RR-0547 2.2 for generic subprograms Like non-generic subprograms, allow merge of specification/body
RR-0604 2.2 for generic subprograms Like non-generic subprograms, allow merge of specification/body
RR-0740 2.2 For optimization with respect to inlined subprograms, allow merging of scopes
RR-0128 4.1 Provide subprograms as parameters to subprograms and entries
RR-0033B 4.4 Need to pass exceptions to subprograms and generic units
RR-0069 4.3 without modifying the original package Allow subprograms and types to be added to a package
RR-0430B 4.1 Need to pass subprograms as parameters
RR-0774K 4.1 Allow subprograms as parameters
RR-0422 4.1 Allow subprograms as parameters and maybe also as values
RR-0512 4.1 Provide subprograms as parameters to subprograms
RR-0128 4.1 Provide subprograms as parameters to subprograms and entries
RR-0641 4.1 Add subprograms as parameters to the language
RR-0060 2.2.9 Allow inlining of subprograms from some but not all call sites
RR-0014 4.1 Need to call subprograms loaded in ROM
RR-0543 2.2.7 Allow accept statements in subprograms nested inside tasks
RR-0269 13.6 Make subprograms not recursive by default
RR-0555 12.2.3 Need ‘‘selective’’ USE clause to get just operators and subprograms of a type
RR-0448 4.3 Allow different sets of subprograms to depend on common declarations
RR-0479 13.1 Need standard subprograms to get user-interface information from OS

RR-0580 2.2.7 Allow accepts within subprograms/packages nested inside tasks
RR-0663 4.2 Allow certain overloading of := and subscripting
RR-0435 9.3 Need secondary standard for simple Ada subset for safety-critical applications
RR-0558 13.4 Deriver of type should be able to hide subset of derived operations
RR-0630 13.2 Due to high implementation costs, define/allow Ada subsets
RR-0734 12.3.11 Generalize cases that allow implicit subtype conversion
AI-00378 12.2.3 Enumeration literals should be made directly visible by a subtype declaration
AI-00390 12.2.3 Character literals should be made directly visible by a subtype declaration
AI-00480 12.2.3 Operators should be made directly visible by a subtype declaration
AI-00540 12.4.2 Completing a private type declaration with a subtype declaration
RR-0690 12.4.2 Allow incomplete and private types to be completed by subtype declaration
RR-0366 13.6 Subtype natural should not include zero
RR-0652 12.2.3 directly visible Declaring a subtype should make the equality operator
AI-00427 13.7 Semi-constrained subtypes
RR-0036 4.5 under a single name by allowing exception subtypes Allow exceptions to be grouped
RR-0425 13.1 Need open ranges in declarations of real subtypes
RR-0610 2.2 Why not allow RENAMES for types and subtypes?
RR-0603 13.5.3 Allow discontiguous subtypes of discrete types
RR-0473 13.5 Allow ‘‘partially’’ constrained subtypes of discriminated records
RR-0058 13.5.3 Allow discontiguous subtypes of enumeration types
AI-00521 13.3 Fixed point subtypes should not inherit SMALL
RR-0584 4.4 Need stricter checking of formal generic subtypes when an instantiation is given
RR-0402 4.3 Need unique hierarchical pathnames for subunit
RR-0262 13.7 Do not require existence of subunit for body stubs
RR-0557 4.3 helps get around the inability to overload subunit names /to provide subprogram bodies
RR-0581C 13.4 context clause/ Allow a pragma ELABORATE for a subunit to mention a package name given in the
AI-00458 4.3 Problem with naming of subunits
RR-0142 4.3 Reduce cases where recompilation of subunits is needed
RR-0154 13.1 compilation unit level Subunits should not have to be at the outermost
RR-0545 13.1 compilation unit level Subunits should not have to be at the outermost
RR-0038 4.3 Allow expanded instead of simple names of subunits to be distinct
RR-0041 4.3 library unit Allow overloaded subunits with respect to a common ancestor
RR-0294 met Chapter 14 optional I/O packages are not suitable for embedded applications; make
RR-0325B 13.6 Allow implementations to experiment with supersets
RR-0437 13.5.3 Provide ‘‘supertype’’ capability for merging enumeration types
RR-0763 2.3 Allow nested scopes to turn off pragma SUPPRESS
RR-0497 13.7 /discriminants for types used as generic actual can yield a surprising run-time error
RR-0436 2.1 Clarify task synchronization point inconsistencies
RR-0119 7.1 composite objects Need synchronized reference to elements of shared
RR-0028 13.4 Add a semicolon terminator to SEPARATE statement syntax
RR-0391 13.4 Clumsy syntax for based numbers, especially in aggregates
RR-0548 13.4 Allow convenient syntax for instantiating a nested generic unit
RR-0753 13.6 Make syntax for task type declarations more consistent
RR-0300 13.2 Use an LR grammar to define the syntax of the language
RR-0326 13.2 Use a different syntax production style
RR-0517 9.3 Provide syntax to declare program units free from side-effects
RR-0518 13.1 Provide syntax to declare subprogram pre/post conditions
RR-0290 13.5 hard to read The syntax used in record representation clauses is
RR-0286B 5.2 interrupts that are also used by the run-time system Embedded system user may need access to
RR-0370A 8.2 declared in library units when reconfiguring a system Can’t recover space
RR-0378 8.1 Need standard means of communication in distributed system
RR-0421D 6.3 may depend inappropriately on the run-time system /ordinary, timed, or conditional calls
RR-0701 13.4 of STANDARD in the same way as for SYSTEM Allow specification
RR-0710 6.3 external events generated by operating system /to tie task entries to asynchronous
RR-0279 2.2 If tasks are not used, the run-time system and compiled code should not include code for/
RR-0175 5.2 between compiler- and target-specific run-time system aspects Define interface
RR-0728 8.1 Need simple Ada run-time system for distributed memory MIMD architectures
RR-0159 4.6 Add standard package of general file system functions
RR-0186 13.3 It is difficult to write an entire operating system in Ada
RR-0176 9.1 Document run-time system performance and memory allocation strategies
RR-0286C 5.2 Run-time system should avoid entering privileged mode
RR-0352 5.1 to return consistently accurate local system time Require Calendar.Clock
RR-0286B 5.2 are also used by the run-time system Embedded system user may need access to interrupts that
RR-0286A 5.2 Embedded system users need the ability to control timer utilities
RR-0068 2.4 that I/O support is optional for embedded systems /Standard should explicitly acknowledge
RR-0224 8.1 Add communication support required for distributed systems
RR-0374 4.2 memory management requirements in distributed systems Ada should address
RR-0376 13.3 in distributed/parallel/multi-processor systems Need special treatment of exceptions

RR-0515 4.2 for specific objects, especially in distributed systems /ability to request indivisible update
RR-0351 13.7 done with it Trusted systems require auto-scrubbing of memory when
RR-0626 6.2 /not portable among compilers, even for the same target machine e.g., because of dope vectors
RR-0554 9.1 Need constraint checks for target of Unchecked_Conversion and I/O input
RR-0476 13.6 functions with the same name as the target type Allow user-written type-conversion
RR-0171 4.3 to be separate from other code Allow target-dependent code (including rep clauses)
RR-0175 5.2 Define interface between compiler- and target-specific run-time system aspects
AI-00450 5.3 Should allow raising of an exception in another task
RR-0380 7.2 Need a task identifier for every task
RR-0487 2.2.11 Need private task entries for exclusive use within the task
RR-0651 5.3 Allow one task to raise an exception in another task
RR-0742 5.3 Need ability to asynchronously stop another task
RR-0013 2.1 than task execution Allow task activation to occur at a higher priority
RR-0394 13.3 Merge concepts of task and package into concept of an object
RR-0711 4.6 I/O by a task in multi- task application should not block whole program
RR-0108 5.1 Need to be able to wake up a task at a particular local time
RR-0774N 4.2 Allow task cleanup on termination of parent
RR-0183 5.4 Asynchronous inter- task communication is not available
RR-0133 7.2 Allow a task component of an array to get its index
RR-0370E 4.2 an allocator Need to recover space for task control blocks when tasks are created by
RR-0628 2.2.11 Need private task entries
AI-00451 4.4 Task entries as formal parameters to generics
RR-0487 2.2.11 Need private task entries for exclusive use within the task
RR-0710 6.3 generated by operating system Need to tie task entries to asynchronous external events
RR-0090 2.2.11 Allow some task entries to be visible, some not
RR-0056 met Do not remove task entry families
RR-0216 9.3 Require that each task entry have at least one accept statement
RR-0013 2.1 activation to occur at a higher priority than task execution Allow task
RR-0194 12.1.1 Disallow referencing a task from outside its master
RR-0380 7.2 Need a task identifier for every task
RR-0711 4.6 whole program I/O by a task in multi-task application should not block
RR-0114 6.3 Allow an address clause for each task instance, and not just on the type
RR-0334 7.2 Need to specify task parameters giving a task its work domain, e.g., to process part of an/
RR-0286D 6.3 Interrupts should be handled with a procedure model, not a task model
RR-0195 6.3 Need interrupt address per task, not task type
RR-0427 12.1.1 permit a function to return a locally-declared task object Do not
RR-0464 12.3.5 Should be able to set STORAGE_SIZE for task objects as well as types
RR-0421C 6.3 Need to associate interrupts with entries of task objects, not task types
RR-0648 12.3.5 Need to set STORAGE_SIZE on task objects, not task types
RR-0703 12.3.5 Need to specify STORAGE_SIZE on task objects, not task types
RR-0104 12.1.1 Prohibit access to a task outside its master
RR-0334 7.2 e.g., to process part of an/ Need to specify task parameters giving a task its work domain,
RR-0015 5.2 Allow task priorities to control all queuing/select decisions
RR-0075 5.2 Queue entries by task priority or FIFO based on application
RR-0347 5.2 /change priorities under program control; allow task priority to increase as a function of lack/
RR-0016 5.2 Allow user-selectable task scheduling algorithms
RR-0124 5.2 Ensure that code dependent on task scheduling algorithms is portable
RR-0436 2.1 Clarify task synchronization point inconsistencies
RR-0400 2.3 Do not allow a task to die silently on an unhandled exception
RR-0407B 2.3 Do not allow a task to die silently on an unhandled exception
RR-0651 5.3 Allow one task to raise an exception in another task
RR-0768 5.3 Need to asynchronously interrupt another task to stop it
RR-0195 6.3 Need interrupt address per task, not task type
RR-0753 13.6 Make syntax for task type declarations more consistent
AI-00570 12.1.1 Releasing heap storage associated with task type instances
RR-0421C 6.3 interrupts with entries of task objects, not task types Need to associate
RR-0648 12.3.5 Need to set STORAGE_SIZE on task objects, not task types
RR-0703 12.3.5 Need to specify STORAGE_SIZE on task objects, not task types
RR-0486 4.4 Allow generic formal task types as well as generic formal limited types
RR-0774B 13.6 Tasking defined as a standard package of functions
RR-0078 5 Ada tasking is too complex, inflexible and inefficient
RR-0278 5.2 disciplines more easily Tasking model should support common scheduling
RR-0279 2.2 and compiled code should not include code for tasking support /not used, the run-time system
AI-00214 2.2.7 Allow accept statements in program units nested in tasks
AI-00453 12.3.5 STORAGE_SIZE for tasks
RR-0021 5.2 Need priority inheritance for server tasks
RR-0023 2.1 Require TERMINATE alternative to terminate library tasks
RR-0173 13.3 with a higher-level entity, i.e., a set of tasks Allow a rendezvous

RR-0203	4.2	Allow finalization code for packages and	tasks
RR-0370B	8.2	Can’t restart library level	tasks
RR-0410	5.1	Provide explicit language support for periodic	tasks
RR-0431	5.3	alternative cannot be used to stop cyclic	tasks A terminate
RR-0521	5.2	support for use of shared memory among	tasks Need more convenient
RR-0543	2.2.7	Allow accept statements in subprograms nested inside	tasks
RR-0580	2.2.7	within subprograms/packages nested inside	tasks Allow accepts
RR-0587	5.4	Provide for communication between loosely coupled	tasks
RR-0370E	4.2	Need to recover space for task control blocks when	tasks are created by an allocator
RR-0279	2.2	compiled code should not include code for/ If	tasks are not used, the run-time system and
RR-0123	7.2	Provide initialization values to	tasks at startup
RR-0370C	2.1	Library level	tasks can’t terminate
RR-0215	2.1	Clarify termination of	tasks dependent on library packages
RR-0370D	5.2	Need to set priorities of	tasks during mode shifts
RR-0063	5.3	critical functions Protect	tasks from being aborted while performing
RR-0037	5.2	time rather than a real-time clock Allow	tasks (i.e., delays) to execute using simulated
RR-0686	6.3	Priority of interrupts higher than normal	tasks is ill-conceived
RR-0084	5.2	Specify standard conventions for using	tasks that permit high-performance/
RR-0771	9.3	Require	tasks to have an accept for each entry
RR-0661	8.2	Need language features for assigning	tasks to nodes
RR-0496	2.1	Clarify termination of	tasks whose masters are library units
RR-0281	2.1	Confusing treatment of	term ‘‘delay statement’’
RR-0235	4.6	Need support for interactive	terminal input/output
RR-0164	4.6	Provide multitasking	terminal I/O in TEXT_IO
RR-0370C	2.1	Library level tasks can’t	terminate
RR-0079	13.6	rarely used	TERMINATE alternative adds little value and is
RR-0431	5.3	cyclic tasks A	terminate alternative cannot be used to stop
RR-0023	2.1	Require	TERMINATE alternative to terminate library tasks
RR-0612	13.3	Should allow both delay and	terminate alternatives in selective wait
RR-0023	2.1	Require TERMINATE alternative to	terminate library tasks
RR-0774N	4.2	Allow task cleanup on	termination of parent
RR-0215	2.1	Clarify	termination of tasks dependent on library packages
RR-0496	2.1	Clarify	termination of tasks whose masters are library units
RR-0028	13.4	Add a semicolon	terminator to SEPARATE statement syntax
AI-00605	4.6	inconsistent with other GET/ GET_LINE skips	terminators at the end of the line, which is
AI-00488	4.6	interactive I/O Skipping of leading line	terminators in GET routines causes problems in
AI-00329	4.6	Look-ahead operation for	TEXT_IO
RR-0164	4.6	Provide multitasking terminal I/O in	TEXT_IO
RR-0360	10.4	Add picture-formatting capabilities to	TEXT_IO
RR-0208	13.4	without waiting for/ Need ability to initiate	TEXT_IO, DIRECT_IO, and SEQ_IO operations
RR-0333	13.3	More precise definition of	TEXT_IO is needed, less implementation freedom
RR-0484	4.6	Add DEFAULT_xy functionality as parameters to generic	TEXT_IO packages
RR-0207	4.6	Add	TEXT_IO support with Exists function and Append/
RR-0551	4.6	Need assignment capability for	TEXT_IO.FILE_TYPE
RR-0047	4.6	Add	TEXT_IO.GET functions
RR-0295	4.6	(make like PUT) Create	TEXT_IO.PUT_LINE for types other than string
RR-0265	13.7	to short-circuit in general, forget AND	THEN Allow implementations
RR-0514	7.3	Provide support for simple parallel	threads within a program unit
RR-0257	2.1	Ensure that BOOLEAN and BYTE arrays can be	tightly packed
RR-0158	13.3	Allow multi-way conditional and	timed entry calls
RR-0656	5.1	Need	timed exceptions for deadline scheduling
RR-0421D	6.3	The treatment of interrupts as ordinary,	timed, or conditional calls may depend/
RR-0286A	5.2	Embedded system users need the ability to control	timer utilities
AI-00519	2.2	Default SMALL should be a power of two	times the range
RR-0276	5.1	specified accuracy and precision control over	timing Need user
RR-0107	5.1	Allow application to specify clock	timing interval if hardware allows this flexibility
RR-0280	5.1	/delays are too inefficient; Calendar time unnecessary;	timing performance must be documented
RR-0368B	4.3	Ensure the library can be manipulated by	tools other than those provided by the compiler/
RR-0665A	5.4	Support multicast message	transfer
RR-0106	5.3	Provide asynchronous	transfer of control
RR-0083	5.3	wait construct Provide asynchronous	transfer of control via entry call/selective
RR-0004	12.2.1	Pragma ELABORATE should be	transitive
RR-0233	12.2.1	Pragma ELABORATE should be	transitive
RR-0773	6.2	variable-length records into a block for data	transmission Need to pack
RR-0017	6.2	Be able to	treat an Ada object as an array of storage units
RR-0699	13.3	an error Do not	treat an unaccepted length clause for a type as
RR-0376	13.3	Need special	treatment of exceptions in/
RR-0421D	6.3	conditional calls may depend/ The	treatment of interrupts as ordinary, timed, or

RR-0179 6.3 The treatment of interrupts is too implementation-dependent
RR-0193 5.2 /priority queues, priority inheritance, and prioritized treatment of open select alternatives
RR-0281 2.1 Confusing treatment of term ‘‘delay statement’’
RR-0719 11.1 Need standard for trig functions, sqrt, etc
RR-0348 11.1 Need predefined functions for real numbers, e.g., trig, log, etc
RR-0358 11.1 Need support for floor, ceiling, truncate, and whole operations
RR-0552 13.4 Need ‘‘padded’’ line input with truncation and pad-fill to ’LENGTH
RR-0252B 11.1 /needs to know/control whether rounding or truncation is used in real calculations
RR-0478 13.1 facilities for restricting use of resources to trusted packages Add language
RR-0351 13.7 memory when done with it Trusted systems require auto-scrubbing of
AI-00327 2.2.5 Instantiating with an incomplete private type
AI-00538 13.4 Declaring constant arrays with an anonymous type
AI-00681 13.4 Can’t declare a constant of a NULL record type
RR-0010 2.2 private type with discriminants to be a derived type Allow the full declaration of a
RR-0114 6.3 for each task instance, and not just on the type Allow an address clause
RR-0129 2.2.2 to be specified for any non-limited type Allow default initialization
RR-0131 13.4 of the enumeration literals of the qualifying type /expression, should have visibility
RR-0161 2.2.2 Allow default initialization for any non-limited type
RR-0166 13.3 the literal representations of an abstract data type Allow definition of
RR-0184 4.2 assignment operator for limited private type Need user-defined
RR-0195 6.3 Need interrupt address per task, not task type
RR-0430A 4.1 Need objects of a subprogram ‘‘type’’
RR-0474 12.2.3 to just enumeration literals and operators of a type Need direct visibility
RR-0476 13.6 functions with the same name as the target type Allow user-written type-conversion
RR-0513 12.3.9 of = for any type, e.g., returning an array type Allow overloading
RR-0551 4.6 Need assignment capability for TEXT_IO.FILE_ TYPE
RR-0555 12.2.3 to get just operators and subprograms of a type Need ‘‘selective’’ USE clause
RR-0577 2.2 a component of an incompletely declared private type /constant of composite type having
RR-0679 13.4 Allow component selection on objects of a private type
RR-0732 2.4 of instantiating ENUMERATION_IO with an integer type Clarify semantics
RR-0302 2.4 The language should define literals for values of type ADDRESS
RR-0467 12.2.3 Need convenient way to rename a type and get its operations
RR-0229 13.4 Need to hide the range of a scalar type and the initial value of an object to ensure/
RR-0699 13.3 Do not treat an unaccepted length clause for a type as an error
RR-0542 2.2.5 One way or another allow usage of private type before its completion declaration
RR-0239B 2.2.3 A renamed type cannot be used in an actual parameter type conversion
RR-0531 4.3 variant record approach Variants of a type can’t be usefully supported with current
AI-00420 3.1 Allow 256 values for type CHARACTER
RR-0239B 2.2.3 A renamed type cannot be used in an actual parameter type conversion
AI-00873 2.3 scalar values Type conversion/qualification of undefined
RR-0510 2.2.10 Re-indexing arrays via type conversions
RR-0715 2.2 Allow user-defined type conversions and attributes for numeric types
RR-0099 12.3.6 Explicit type conversions should be allowed in static expressions
AI-00540 12.4.2 Completing a private type declaration with a subtype declaration
RR-0677 2.2.2 Allow initialization clauses on scalar type declarations
RR-0259 13.7 Incomplete type declarations are dangerous and unnecessary
RR-0753 13.6 Make syntax for task type declarations more consistent
AI-00518 13.4 Fixed and floating type declarations needlessly different
RR-0191 2.2 model numbers should include the bounds of the type definition Fixed point
RR-0230 2.2.2 Allow initialization to be associated with any type definition
RR-0456 2.2.2 Allow initialization to be associated with a type definition
RR-0506 2.2.2 Allow initialization to be associated with a type definition
RR-0566 2.2 model numbers should include the bounds of the type definition Fixed point
AI-00429 13.4.4 Allow array type definition for record component
RR-0492 11.1 and exponent information in floating point type definitions Decouple mantissa
RR-0336 13.4.4 array-of-array case Allow array type definitions in records; nice for
RR-0513 12.3.9 Allow overloading of = for any type, e.g., returning an array type
RR-0421B 6.3 /different from memory address structure; a single type for both is inappropriate
RR-0680 13.1 Predefined exponentiation should take any integer type for exponent
RR-0627 4.4 Allow partial match to formal type for records
RR-0577 2.2 Allow deferred constant of composite type having a component of an incompletely declared/
RR-0442 4.3 Extend Ada to allow a package type hierarchy
AI-00285 4.4 Need to be able to access a base numeric type in some algorithms
AI-00570 12.1.1 Releasing heap storage associated with task type instances
RR-0450 6.4 Need efficient manipulation of buffers whose type is determined at run time
RR-0462 12.3.7 Allow selected component form of type mark in a formal part even when the selected/
RR-0675 12.3.7 Allow a subprogram identifier to be used as a type mark in its specification
RR-0579 12.3.7 subprogram named FOO Allow a type mark of form P.FOO in the formal part of a

RR-0315 2.4 e.g., INTEGER_32, to improve/ Allow integer type names that indicate representation size,
RR-0009 12.3.6 Allow static conversion to static discrete type of static discrete expression
AI-00479 12.3.10 Initialize access type OUT parameters to null
RR-0395 2.2 Include formal parameter names in parameter/result- type profile
RR-0600 2.2 Allow formal parameter names in parameter/result- type profile
RR-0532 2.2.6 share name Allow same- type record components in different variants to
RR-0558 13.4 Deriver of type should be able to hide subset of derived operations
RR-0252D 2.2 Fixed point type should include the bounds of the range definition
RR-0239A 12.2.3 Renaming an enumeration type should make literals visible
RR-0096C 12.4.2 Allow the full declaration of a private type to be provided by a renaming declaration
RR-0466 4.2 Allow user-defined finalization for objects of a type to ensure release of resources
RR-0523 4.2 Allow user-defined finalization for objects of a type to ensure release of resources
RR-0010 2.2 Allow the full declaration of a private type with discriminants to be a derived type
RR-0392 13.5 Need ‘‘semi-limited’’ type with predefined := but no predefined =
AI-00345 4.6 Record type with variant having no discriminants
RR-0190 4.4 Allow use of a base type within a generic unit
RR-0511 4.4 Allow use of a base type within a generic unit
RR-0476 13.6 the target type Allow user-written type-conversion functions with the same name as
RR-0458 4.4 Need convenient way to escape into weakly typed subprogram call
RR-0596 12.3.13 Allow END type_name to substitute for END RECORD
RR-0673 12.3.13 Allow ‘‘END RECORD’’ type_name to substitute for ‘‘END RECORD’’
AI-00291 4.4 package that works for all floating point types Can’t define a generic
RR-0006 4.4 Distinguish unconstrained/constrained generic formal types
RR-0008 12.3.9 Allow overloading of the equality operator for all types
RR-0025 12.3.9 of the equality operator with different operand types Allow overloading
RR-0048 2.2.4 include representation attributes of composite types Extend static expressions to
RR-0058 13.5.3 Allow discontiguous subtypes of enumeration types
RR-0061 2.4 Make Long_Float and Short_Float required types
RR-0070 4.2 Allow user-defined assignment for limited types
RR-0081 4.1 Provide subprogram and package types
RR-0155 12.3.3 Define RANGE attribute for scalar types
RR-0160 4.2 Allow user-defined assignment for limited types
RR-0167 4.3 Allow classes of abstract data types
RR-0188 6.1 and bit-wise logical operations on integer types /applications need unsigned integers
RR-0304 12.3.3 Define RANGE attribute for scalar types
RR-0313 13.4 constants of arbitrary (i.e., non-private) types Allow deferred
RR-0363 12.3.1 to apply to real types as well as discrete types Allow ’VALUE and ’IMAGE
RR-0406 13.4.1 Allow user-defined attributes for user-defined types
RR-0412 12.3.9 Allow overloaded = for all types, not just limited types
RR-0413 4.2 Allow user-written := for all types
RR-0421C 6.3 with entries of task objects, not task types Need to associate interrupts
RR-0423 2.2 restriction on full declarations of private types Remove discriminant
RR-0433 6.1 There is a need for predefined unsigned integer types
RR-0437 13.5.3 Provide ‘‘supertype’’ capability for merging enumeration types
RR-0446 4.4 constrained/unconstrained generic types /the contract model by distinguishing
RR-0449 13.1 Do not allow unchecked conversion of private types
RR-0454 11.1 Need Entier function or attribute for real types
RR-0460 6.1 Ada needs to provide support for unsigned integer types
RR-0464 12.3.5 to set STORAGE_SIZE for task objects as well as types Should be able
RR-0472 4.4 Distinguish unconstrained/constrained generic formal types
RR-0486 4.4 task types as well as generic formal limited types Allow generic formal
RR-0505A 4.3 Provide extendable record types
RR-0509 13.4.1 attributes for user-defined or private types Allow user-defined
RR-0529 13.5 based on run-time queries about properties of types Allow selection of operations
RR-0530 13.3 Insufficient support for mutants of limited types
RR-0568 2.2 Allow non-nested variant parts in record types
RR-0572 13.1 with respect to all predefined integer types Need predefined operators
RR-0595 2.2.2 Allow default initialization for all types
RR-0603 13.5.3 Allow discontiguous subtypes of discrete types
RR-0609 4.2 Allow user-defined override of =, /=, := on all types
RR-0617 13.6 Eliminate anonymous array types
RR-0622 2.1 use ‘‘metatype’’ in describing generic formal types The Standard should
RR-0648 12.3.5 Need to set STORAGE_SIZE on task objects, not task types
RR-0660 4.2 Need constructors and destructors for package types
RR-0664 12.3.1 Need ’IMAGE and ’VALUE attributes for floating-point types
RR-0672 13.4 Need anonymous pointer types
RR-0703 12.3.5 specify STORAGE_SIZE on task objects, not task types Need to
RR-0715 2.2 type conversions and attributes for numeric types Allow user-defined

RR-0716 11.1 Unify and add attributes for numeric types
RR-0722 4.4 Need generic formal record types
RR-0741 7.3 Need hot performance on vector machines; add vector types and operands
RR-0414 4.1 Ada needs subprogram types and subprogram objects
RR-0610 2.2 Why not allow RENAMES for types and subtypes?
RR-0563 4.1 Need to allow subprogram types and variables
RR-0080 13.3 Derived types are clumsy
RR-0272 13.6 Limited types are of little true value
RR-0122 2.2 Permit an implementation to reject some integer types as array indexes
AI-00452 4.4 Allow record types as generic formal parameters
RR-0443 13.4.4 Need for anonymous array types as record components
RR-0363 12.3.1 Allow 'VALUE and 'IMAGE to apply to real types as well as discrete types
RR-0486 4.4 Allow generic formal task types as well as generic formal limited types
RR-0172 4.3 Make import and export of types easier
RR-0503 4.1 Provide subprogram types for dispatcher-style programming
RR-0629 4.1 Need procedure and function types for use in subprogram calls
RR-0533 4.3 Mutually recursive types from different packages cannot be done
RR-0482 4.3 Multiple derived types from same package do not generate needed operations
RR-0052 4.3 Multiple derived types from same package do not give desired operations
AI-00404 2.2 Use of incomplete private types in generic formal part
RR-0273 13.6 There are problems with private types in the language
RR-0389 13.4 There is a need for ''cyclic'' discrete types in the language
RR-0082 2.2.5 Allow declaration of objects of private types in visible package specification
RR-0320 13.5 Generalize case statement for other types, including REAL
RR-0549 4.4 Ensure the use of unconstrained actual types is always legal
RR-0012 13.3 Mutation of types is needed for AI applications
RR-0001 4.2 Limited types need assignment, constants
RR-0418 2.2 Representation clauses for array types need to be added
RR-0733 13.5 Need fixed-point types not centered on zero
RR-0412 12.3.9 Allow overloaded = for all types, not just limited types
RR-0649 2.2.2 Allow default initialization for all types (not just records)
RR-0110 6.4 placement of and access to data in different types or regions of memory /control over
RR-0098 13.4 Generalize incomplete typing for types other than access or private
RR-0295 4.6 Create TEXT_IO.PUT_LINE for types other than string (make like PUT)
RR-0197 13.6 designated object cannot be/ For access types, parameter mode IN should mean the
RR-0287 2.4 Make access types point directly to designated object
RR-0684 4.3 Related packages need access to a private type's representation
RR-0560 4.3 Need to access a private type's representation in related packages
RR-0578 2.2.3 Out-mode parameters of limited private types should be allowed
RR-0007 2.4 Default representation for enumeration types should be specified
RR-0202 4.2 Relax parameter mode rules for limited types that have an assignment operation
RR-0069 4.3 modifying the original/ Allow subprograms and types to be added to a package without
RR-0690 12.4.2 Allow incomplete and private types to be completed by subtype declaration
RR-0668 4.3 Need package types to get, for example, an array of packages
RR-0019 4.2 safely controlling use of collections Allow types to specify finalization procedures for
RR-0497 13.7 Presence of default discriminants for types used as generic actual can yield a/
RR-0611 4.1 Allow subprogram types, variables, constants, parameters, etc
RR-0599 4.3 Certain changes to derived/private types will help inheritance
RR-0712 4.4 Need ability to declare double precision numeric types within a generic unit
RR-0098 13.4 Generalize incomplete typing for types other than access or private
RR-0103A 2.2.3 Allow unchecked conversion for IN OUT and OUT parameters
RR-0449 13.1 Do not allow unchecked conversion of private types
RR-0353 2.4 compiler-dependent fields Unchecked conversion should eliminate
RR-0554 9.1 Need constraint checks for target of Unchecked_Conversion and I/O input
RR-0549 4.4 Ensure the use of unconstrained actual types is always legal
RR-0446 4.4 contract model by distinguishing constrained/ unconstrained generic types Tighten the
RR-0006 4.4 Distinguish unconstrained/constrained generic formal types
RR-0472 4.4 Distinguish unconstrained/constrained generic formal types
AI-00873 2.3 Type conversion/qualification of undefined scalar values
RR-0126 13.4 Allow underscore before ''E'' in exponents
RR-0400 2.3 Do not allow a task to die silently on an unhandled exception
RR-0407B 2.3 Do not allow a task to die silently on an unhandled exception
RR-0581 12.2.1 of pragma ELABORATE are error-prone and unhelpful Rules specifying the position
RR-0041 4.3 with respect to a common ancestor library unit Allow overloaded subunits
RR-0190 4.4 Allow use of a base type within a generic unit
RR-0511 4.4 Allow use of a base type within a generic unit
RR-0514 7.3 for simple parallel threads within a program unit Provide support
RR-0548 13.4 syntax for instantiating a nested generic unit Allow convenient

RR-0581C 13.4 given in the context clause of a parent library unit /for a subunit to mention a package name
RR-0712 4.4 double precision numeric types within a generic unit Need ability to declare
RR-0757 2.1 Clean up definitions of program unit and compilation unit
RR-0769 2.1 Correct wording in the definition of ancestor unit
RR-0757 2.1 Clean up definitions of program unit and compilation unit
RR-0483 12.3.7 /to have the same identifier as the generic unit (as is allowed for package instances)
RR-0218 12.2.1 Make the implementation find a good library- unit elaboration order
RR-0396 12.2.1 for pragma ELABORATE Add library unit elaboration ordering rules to reduce need
RR-0154 13.1 not have to be at the outermost compilation unit level Subunits should
RR-0545 13.1 not have to be at the outermost compilation unit level Subunits should
RR-0586 4.4 Different instantiations of the same generic unit may have to evaluate their actual/
RR-0205 12.3.13 Allow program unit name on PRIVATE, BEGIN, and EXCEPTION
RR-0407A 4.5 Need exception name, line number, and unit name where raised
RR-0065 4.3 pragmas to be separated from the compilation unit to which they apply /clauses and various
RR-0774C 4.3 Extend control of library unit visibility
AI-00850 2.3 Rejecting a unit when a pragma's assumptions are not met
RR-0017 6.2 to treat an Ada object as an array of storage units Be able
RR-0033B 4.4 Need to pass exceptions to subprograms and generic units
RR-0383 4.4 Need generic exceptions for truly reusable generic units
RR-0457 4.3 units as groups, control visibility of library units Structure library
RR-0496 2.1 Clarify termination of tasks whose masters are library units
RR-0698 4.3 portable and non-portable code into separate units Need ability to separate
RR-0101B 4.4 Need to pass exceptions as parameters to generic units and subprograms
RR-0526B 4.4 Need to pass exceptions as parameters to generic units and subprograms
RR-0457 4.3 Structure library units as groups, control visibility of library units
RR-0517 9.3 Provide syntax to declare program units free from side-effects
AI-00214 2.2.7 Allow accept statements in program units nested in tasks
RR-0095 13.4 Allow applicable units to be named in USE clauses and pragma ELABORA/
RR-0607 13.1 Allow names of compilation units to be overloadable, operator symbols
RR-0035 13.5.4 Allow generic units to be overloaded
RR-0370A 8.2 Can't recover space declared in library units when reconfiguring a system
RR-0519 13.1 Simplify overload rules for ambiguous/ universal expressions
RR-0591 2.2.8 Allow fixed-point multiply/divide with universal real operands
RR-0689 12.2.4 Optional bodies should not be unlinked without a warning
RR-0005 4.4 in generic packages make code sharing unnecessarily difficult Exception declarations
RR-0259 13.7 Incomplete type declarations are dangerous and unnecessary
RR-0368A 4.3 Ensure unnecessary recompilation is avoided
RR-0688 4.3 redeclaring a subprogram body Unnecessary recompilation required when
RR-0280 5.1 Short delays are too inefficient; Calendar time unnecessary; timing performance must be/
RR-0284 13.1.1 Machine-code insertions are unreadable; replace with INLINE macros
RR-0390 3.1 Need 8-bit unsigned CHARACTER for Greek and graphics symbols
RR-0187 2.4 Need to allow unsigned enumeration representation specifications
RR-0332 6.1 Provide unsigned integer capability
RR-0433 6.1 There is a need for predefined unsigned integer types
RR-0460 6.1 Ada needs to provide support for unsigned integer types
RR-0138 6.1 Need full-sized unsigned integers
RR-0188 6.1 operations on/ Embedded applications need unsigned integers and bit-wise logical
AI-00600 6.1 Why we need unsigned integers in Ada
RR-0044 13.2 There is no need to add unsigned integers to Ada
RR-0721 6.1 Try to add unsigned integers to the language
RR-0692 2.3 Allow implementation-defined pragmas to cause unsuccessful compilation if restrictions/
RR-0565 2.2 clauses inappropriate 'SMALL is unsuitably defined; need for representation
RR-0615 2.2.12 Define LOOP/ UNTIL control structure as in Pascal
RR-0515 4.2 Need ability to request indivisible update for specific objects, especially in/
RR-0544 4.2 Need indivisible update on reference counts
RR-0502 2.1 The Standard should be consistent in its use of upper and lower cases
AI-00216 10.2 /methods for testing whether characters are numeric, upper case, lower case, control, etc.,/
RR-0643 4.2 Garbage collection can now be done well; encourage its use
RR-0326 13.2 Use a different syntax production style
RR-0300 13.2 Use an LR grammar to define the syntax of the language
RR-0043 13.1.1 Make it easier and more portable to use assembler with Ada
RR-0588 13.4 Provide a form of USE clause that hides outer homographs
RR-0555 12.2.3 subprograms of a type Need ''selective'' USE clause to get just operators and
AI-00274 13.1 Proposed extension of the USE clause — record component visibility
RR-0095 13.4 Allow applicable units to be named in USE clauses and pragma ELABORATE
RR-0100 13.4 Allow constants to use default values to get value
RR-0629 4.1 Need procedure and function types for use in subprogram calls
RR-0034 3.1 Ada should use ISO 8859/1-9 (8-bit) character set

AI-00510	3.1		Use ISO symbols and standards in the Ada ISO Standard
RR-0622	2.1	The Standard should	use ‘‘metatype’’ in describing generic formal types
RR-0190	4.4	Allow	use of a base type within a generic unit
RR-0511	4.4	Allow	use of a base type within a generic unit
RR-0291	6.4	Clarify whether	use of an address clause causes storage to be initialized
AI-00539	2.2.4	representation clauses Allow	use of array/record attributes in
RR-0019	4.2	finalization procedures for safely controlling	use of collections Allow types to specify
RR-0642	13.4	Add label variables to support	use of finite state machines
AI-00404	2.2	formal part	Use of incomplete private types in generic
RR-0252A	11.1	for IEEE floating point standard; allow full	use of machine characteristics Ensure support
RR-0438	3.1	Allow	use of multi-octet character set
RR-0571A	12.2.5	when index bounds are determined by/ Allow	use of OTHERS choice with named associations
RR-0029	12.2.5	index constraint is determined by/ Allow	use of OTHERS with named associations when the
RR-0767	12.2.1	elaboration order problem without requiring the	use of pragma ELABORATE Solve the
RR-0576	13.4	Allow parameter default expressions to make	use of previous IN parameters
RR-0557	4.3	subprogram bodies helps get around the/ The	use of renaming declarations to provide
RR-0478	13.1	Add language facilities for restricting	use of resources to trusted packages
RR-0521	5.2	Need more convenient support for	use of shared memory among tasks
RR-0549	4.4	Ensure the	use of unconstrained actual types is always legal
RR-0502	2.1	The Standard should be consistent in its	use of upper and lower cases
RR-0731	11.1	as a basis for Ada’s floating point model	Use the Language Compatible Arithmetic Standard
RR-0487	2.2.11	Need private task entries for exclusive	use within the task
RR-0531	4.3	Variants of a type can’t be	usefully supported with current variant record approach
RR-0121	5.2	Provide more	user control over scheduling decisions
RR-0737	5.2	a select statement Allow reliable	user control over selection of alternatives in
RR-0444	13.4	can be raised Let the	user limit the places where a given exception
RR-0286B	5.2	also used by the run-time/ Embedded system	user may need access to interrupts that are
RR-0276	5.1	over timing Need	user specified accuracy and precision control
RR-0112	4.2	Provide	user support for controlled space reclamation
RR-0541	4.2	support memory management Allow	user-defined :=, =, DESTROY operations to
RR-0088	4.2	Problems associated with	user-defined assignment
RR-0070	4.2	Allow	user-defined assignment for limited types
RR-0160	4.2	Allow	user-defined assignment for limited types
RR-0184	4.2	private type Need	user-defined assignment operator for limited
RR-0674	13.4.1	Allow	user-defined attributes as functions
RR-0509	13.4.1	private types Allow	user-defined attributes for user-defined or
RR-0406	13.4.1	Allow	user-defined attributes for user-defined types
RR-0613	13.4.1	problems with implementation-defined/	User-defined attributes solve portability
RR-0466	4.2	to ensure release of resources Allow	user-defined finalization for objects of a type
RR-0523	4.2	to ensure release of resources Allow	user-defined finalization for objects of a type
RR-0509	13.4.1	Allow user-defined attributes for	user-defined or private types
RR-0682	13.5.1	‘‘:-’’, etc Allow	user-defined overloaded operators such as ‘‘?’’,
RR-0609	4.2	Allow	user-defined override of =, /=, := on all types
RR-0475	4.2	Need automatically-invoked	user-defined routines to reclaim storage
RR-0715	2.2	for numeric types Allow	user-defined type conversions and attributes
RR-0406	13.4.1	Allow user-defined attributes for	user-defined types
RR-0479	13.1	Need standard subprograms to get	user-interface information from OS
RR-0337	5.2	Provide some form of	user-modifiable priorities
RR-0116	5.2	change and graceful degradation	User-modifiable priorities needed for mode
RR-0426A	12.2.4	The effect of an optional package body is confusing to	users
RR-0286A	5.2	Embedded system	users need the ability to control timer utilities
RR-0120	4.2	when space is exhausted Allow	users to defer the signalling of STORAGE_ERROR
RR-0248	13.1	that are outside record values Allow	users to specify locations for discriminants
RR-0016	5.2	Allow	user-selectable task scheduling algorithms
RR-0092	4.2	Allow	user-specified finalization
RR-0118	4.2	STORAGE_ERROR recovery Provide a	user-specified storage reserve for
RR-0413	4.2	Allow	user-written := for all types
RR-0669	4.2	Allow	user-written := routines
RR-0476	13.6	same name as the target type Allow	user-written type-conversion functions with the
RR-0328	9.3	Require compilers to report questionable	uses of the language
RR-0329	13.1		Using a deferred constant before it has a value
RR-0647	4.1	to select actions depending on state without	using case statements Need ability
RR-0037	5.2	Allow tasks (i.e., delays) to execute	using simulated time rather than a real-time clock
RR-0084	5.2	Specify standard conventions for	using tasks that permit high-performance/
RR-0286A	5.2	Embedded system users need the ability to control timer	utilities
RR-0059	2.2.14	for returning a representation’s underlying	value Need an attribute
RR-0097	13.4	Allow/require explicit action to get default parameter	value
RR-0100	13.4	Allow constants to use default values to get	value

RR-0272 13.6 Limited types are of little true value
RR-0329 13.1 Using a deferred constant before it has a value
RR-0453 11.1 or attribute yielding the sign of a numeric value Provide a special function
RR-0567 2.2 declaration to get constraints from initial value Allow variable
RR-0234 12.1.2 ‘‘Sub-null’’ ranges are of little value and an implementation burden
RR-0363 12.3.1 as discrete types Allow ’ VALUE and ’IMAGE to apply to real types as well
RR-0079 13.6 TERMINATE alternative adds little value and is rarely used
RR-0465 2.2.14 to get the representation from an enumeration value and vice versa Need a way
RR-0664 12.3.1 Need ’IMAGE and ’ VALUE attributes for floating-point types
RR-0614 12.3.4 condition RETURN to make selection of returned value clearer Allow WHEN
RR-0031 13.5.3 Provide a way to test for a value in a non-contiguous set
RR-0341 2.2 Allow discriminant value in record aggregate to be non-static
RR-0653 8.2 Need to declare constants whose value is supplied after linking
RR-0621C 13.4 Allow case statements to dispatch on value of an exception
RR-0229 13.4 /hide the range of a scalar type and the initial value of an object to ensure these values are/
RR-0255 11.1 Provide a function for returning the value of the next floating point number
AI-00873 2.3 Type conversion/qualification of undefined scalar values
RR-0040 2.2.14 to determine the internal coding of enumeration values Need a way
RR-0169 13.4 for actual or default generic formal subprogram values Allow ‘‘null’’ procedures
RR-0220 2.2.14 the internal code associated with enumeration values Need way to get
RR-0248 13.1 for discriminants that are outside record values Allow users to specify locations
RR-0338 6.4 conversion between ADDRESS values and access values /pointers to static objects and safe
RR-0350 2.1 Clarify wording dealing with default initial values
RR-0422 4.1 Allow subprograms as parameters and maybe also as values
RR-0338 6.4 objects and safe conversion between ADDRESS values and access values /pointers to static
RR-0229 13.4 /the initial value of an object to ensure these values are not used directly by programmers
AI-00874 6.4 Ensure that access values are values of ’ADDRESS
AI-00420 3.1 Allow 256 values for type CHARACTER
RR-0749 12.3.11 for slices serving as actual parameters and as values in record components /index sliding
AI-00874 6.4 Ensure that access values are values of ’ADDRESS
RR-0302 2.4 The language should define literals for values of type ADDRESS
RR-0258 6.4 Need access values that point to declared objects
RR-0238 6.4 Allow access values to designate read-only memory
RR-0100 13.4 Allow constants to use default values to get value
RR-0293 6.4 Allow access values to point to declared objects
RR-0123 7.2 Provide initialization values to tasks at startup
RR-0567 2.2 initial value Allow variable declaration to get constraints from
RR-0032 12.2.2 Allow grouping of variable declarations and related subprograms
RR-0054 13.2 Do not add variable length strings to the language
RR-0773 6.2 Need to pack variable-length records into a block for data transmission
RR-0163 10.4 equality and assignment/ Need support for variable-length strings with appropriate
RR-0563 4.1 Need to allow subprogram types and variables
RR-0247 13.6 Don’t initialize access variables by default to NULL
RR-0611 4.1 Allow subprogram types, variables, constants, parameters, etc
RR-0130 4.6 Replace DEFAULT_xy variables in Chapter 14 by functions
RR-0642 13.4 Add label variables to support use of finite state machines
RR-0271 13.6 Distinguish storage classes for variables with key words like CONTROLLED or STATIC
RR-0018 6.4 Need pre-elaborated constant arrays with variable-sized elements
AI-00345 4.6 Record type with variant having no discriminants
RR-0568 2.2 Allow non-nested variant parts in record types
RR-0531 4.3 a type can’t be usefully supported with current variant record approach Variants of
RR-0593 4.6 Mandate implementation of variant record I/O in DIRECT_IO/SEQUENTIAL_IO
RR-0707 2.2.6 Need same-name component identifiers in different variants
RR-0531 4.3 with current variant record approach Variants of a type can’t be usefully supported
RR-0532 2.2.6 Allow same-type record components in different variants to share name
RR-0327 10.4 Add varying length strings to the language
RR-0419 10.4 Add some form of support for varying length strings to the language
RR-0640 6.1 Need to access chunk of a bit vector as a whole
RR-0741 7.3 Need hot performance on vector machines; add vector types and operands
RR-0738 7.3 Add facilities to support vector processing hardware
RR-0741 7.3 Need hot performance on vector machines; add vector types and operands
RR-0626 6.2 the same target machine e.g., because of dope vectors /not portable among compilers, even for
RR-0368B 4.3 tools other than those provided by the compiler vendor /the library can be manipulated by
RR-0759 13.3 Add real-time and verification facilities for control engineering
RR-0289 6.2 discriminant is present Need multiple views of a record structure even when no
AI-00274 13.1 of the USE clause — record component visibility Proposed extension
RR-0774C 4.3 Extend control of library unit visibility
RR-0624 12.2.3 Provide selective direct visibility into a package

RR-0182	8.1	running on different processors Define	visibility limits for parts of a program
RR-0393	12.2.3	by renaming Can’t get direct	visibility of fixed point mult and div operator
RR-0457	4.3	Structure library units as groups, control	visibility of library units
RR-0073	4.3	program library Allow	visibility of names to be restricted within a
RR-0022	12.2.3	Need direct	visibility of operators declared in another package
RR-0232	12.2.3	Need to allow direct	visibility of operators in packages
RR-0727	12.2.3	Need selective direct	visibility of package declarations
RR-0131	13.4	In a qualified expression, should have	visibility of the enumeration literals of the/
RR-0274	2.1	The	visibility rules could be explained more clearly
RR-0057	12.2.3	Need direct	visibility to infix operators in another package
RR-0474	12.2.3	operators of a type Need direct	visibility to just enumeration literals and
RR-0694	12.2.3	Need easy direct	visibility to the equality operations
RR-0239A	12.2.3	Renaming an enumeration type should make literals	visible
RR-0429	12.2.3	makes just overloadable declarations directly	visible Need construct that
RR-0652	12.2.3	should make the equality operator directly	visible Declaring a subtype
AI-00378	12.2.3	Enumeration literals should be made directly	visible by a subtype declaration
AI-00390	12.2.3	Character literals should be made directly	visible by a subtype declaration
AI-00480	12.2.3	Operators should be made directly	visible by a subtype declaration
RR-0082	2.2.5	Allow declaration of objects of private types in	visible package specification
RR-0090	2.2.11	Allow some task entries to be	visible, some not
RR-0678	7.1	for data shared between programs; need	VOLATILE Pragma SHARED is not sufficient
RR-0434	7.1	Need atomic read/write operations on shared	volatile memory
RR-0415	5.2	entry-queues, and prioritized selective	wait Allow priority inheritance, prioritized
RR-0612	13.3	delay and terminate alternatives in selective	wait Should allow both
RR-0697	5.2	Allow entry call alternative in selective	wait
RR-0083	5.3	transfer of control via entry call/selective	wait construct Provide asynchronous
RR-0498	5.2	statements and entry calls Make selective	wait symmetrical with respect to accept
RR-0208	13.4	DIRECT_IO, and SEQ_IO operations without	waiting for completion /to initiate TEXT_IO,
RR-0108	5.1	Need to be able to	wake up a task at a particular local time
RR-0689	12.2.4	Optional bodies should not be unlinked without a	warning
RR-0261	2.3	Need compile-time	warnings for access before elaboration errors
RR-0242	2.3	Require compilation	warnings for potential run-time errors
RR-0754	2.3	Require	warnings for unrecognized pragmas
RR-0756	2.3	Require	warnings when pragmas are ignored
RR-0458	4.4	Need convenient way to escape into	weakly typed subprogram call
RR-0747	13.6	Provide better support for ‘‘light-’’	weight parallelism (as in Linda)
RR-0765	13.1	Allow	‘‘when Package_Name.others =>’’ as exception handler
RR-0236	2.4	/dependent behavior, or at least, ensure it is documented	whenever possible
AI-00216	10.2	Provide standard methods for testing	whether characters are numeric, upper case, lower/
RR-0252B	11.1	Programmer needs to know/control	whether rounding or truncation is used in real calculations
RR-0291	6.4	to be initialized Clarify	whether use of an address clause causes storage
RR-0063	5.3	Protect tasks from being aborted	while performing critical functions
RR-0180	4.1	There is a need for procedures as parameters for X-	Windows, etc
RR-0774I	13.1	Create separate standards, such as X-	Windows, SQL
RR-0689	12.2.4	Optional bodies should not be unlinked	without a warning
RR-0069	4.3	Allow subprograms and types to be added to a package	without modifying the original package
RR-0767	12.2.1	Solve the elaboration order problem	without requiring the use of pragma ELABORATE
RR-0647	4.1	Need ability to select actions depending on state	without using case statements
RR-0208	13.4	/to initiate TEXT_IO, DIRECT_IO, and SEQ_IO operations	without waiting for completion
AI-00291	4.4	Can’t define a generic package that	works for all floating point types
RR-0633	6.1	Provide logical operations (e.g.,	XOR) for integers
RR-0497	13.7	/for types used as generic actual can	yield a surprising run-time error
RR-0453	11.1	Provide a special function or attribute	yielding the sign of a numeric value
RR-0366	13.6	Subtype natural should not include	zero
RR-0733	13.5	Need fixed-point types not centered on	zero
RR-0637	11.1	Ada programs should run as though negative	zero did not exist
AI-00442	13.4	Time	zone information in package CALENDAR