MAXAda

Version 3.3.3 Release Notes
(PowerMAX)

August 2003

0890516-3.3.3

READ ME BEFORE
INSTALLING
THIS PRODUCT

CONCURRENT
COMPUTER
CORPORATION

Contents

1.0. Introduction

This document provides an overview of Version 3.3.3 of the MAXAda™ Compilation System on the PowerMAX™ operating system. MAXAda 3.3.3 is a maintenance release containing all patches and development updates to MAXAda 3.3.2 since its release. See “Changes in This Release” on page 9 for a detailed listing of those changes.

MAXAda supports development of Ada95 programs running under Concurrent Computer Corporation’s PowerMAX. MAXAda processes the Ada language as specified by the *Reference Manual for the Ada Programming Language, ANSI/ISO/IEC-8652:1995*, referred to in this document as the *Ada 95 Reference Manual* or RM.

MAXAda 3.3.3 is based on MAXAda 3.1 which has been validated using Version 2.1 of the Ada Conformity Assessment Test Suite (certificate #A981215E2.1-047).

In addition, MAXAda 3.3.3 includes POSIX® 1003.5, a complete implementation of the Institute of Electrical and Electronic Engineers (IEEE) standard IEEE-Std-1003.5-1992.

MAXAda works in conjunction with NightBench™, a program development environment that brings together MAXAda with a collection of tools crucial to Ada program development under a graphical user interface. NightBench aids the user by handling the intricate details involved in consistent program generation.

MAXAda also works in conjunction with the NightView™ Source-Level Debugger, a non-intrusive GUI monitoring and debugging tool specifically designed for real-time.

Optional products that are not bundled with the MAXAda product are available as stand-alone products that must be obtained separately. Optional products that complement MAXAda include the following:

- ID Tools (Ada and C cross-referencing tools)
- NightSim™ (a graphical real-time scheduling tool)
- NightTrace™ (a graphical real-time trace and analysis tool)
- AXI™ (an Ada X interface to the full Xlib, Xt, and Motif libraries)

For more information about optional Ada products and support tools, contact the Concurrent Software Support Center at the number listed in the “Documentation” section.

2.0. Documentation

Table 2-1 lists the MAXAda 3.3.3 documentation available from Concurrent.

Table 2-1. MAXAda Version 3.3.3 Documentation

Manual Name	Pub. Number
MAXAda Reference Manual	0890516-100
MAXAda Version 3.3.3 Release Notes (PowerMAX)	0890516-3.3.3

Copies of the Concurrent documentation can be ordered by contacting the Concurrent Software Support Center. The toll-free number for calls within the continental United States is 1-800-245-6453. For calls outside the continental United States, the number is 1-954-283-1822 or 1-305-931-2408.

Additionally, the manuals listed above are available:

- online using the X Window System utility, **`nhelp`**
- on the Concurrent Computer Corporation web site at www.ccur.com

3.0. Prerequisites

Prerequisites for MAXAda Version 3.3.3 are as follows:

3.1. Software

Minimal Requirements

- PowerMAX OS 4.3 or later
- Élan License Manager™ 5.0.2

Recommended Software

- NightView™ Version 5.6
- NightBench™ Version 2.3
- X Window System™ (X11 Version 6.4.2) (*for graphical user interfaces*)

3.2. Hardware

- Computer Systems:

  Power Hawk™ 620 and 640

  Power Hawk 710, 720 and 740

  Power Hawk 910 and 920

  PowerStack™ II and III

  Night Hawk® Series 6000

  TurboHawk™

  PowerMAXION™
- Board-Level Products:

  Motorola MVME2604

  Motorola MVME4604
- 64MB physical memory (absolute minimum configuration)
- 128MB physical memory (recommended minimum configuration)
- 32MB physical memory per MAXAda user (minimum configuration)
- 1GB swap space (recommended minimum configuration)
- X Window System display terminal (optional)

4.0. System Installation

The MAXAda product is installed as a standard PowerMAX software package and utilizes the standard PowerMAX product installation mechanism, **`pkgadd`** (see **`pkgadd(1)`**).

The package name is **`MAXAda`**. This name is case-sensitive.

NOTE

If you are installing this version of MAXAda on a machine that has an earlier version installed, see "Multiple Releases".

Please refer to the "Installing Add-on Software" chapter in the *System Administration Volume I* (0890429) manual for instructions on software installation.

After installation of the package, the default MAXAda release on the system will be MAXAda 3.3.3 (denoted as *phase3.3.3* by the toolset). The system administrator can change the default release using the **`a.install`** tool (see the "MAXAda Utilities" chapter of the *MAXAda Reference Manual* (0890516) for more information).

If a previous version of MAXAda was already installed on the system, existing MAXAda environments will not be affected by the installation of this new release. Development in those existing environments may proceed while continuing to use the **previous** release.

All newly created environments will use MAXAda 3.3.3 unless otherwise specified by the user (or the system administrator has changed the default MAXAda version to something other than 3.3.3).

Users must completely remove their existing environments via the **`a.rmenv`** command and recreate them with the **`a.mkenv`** command if they wish to use the new version.

4.1. Multiple Releases

MAXAda supports multiple releases on a particular machine. However, the installation process is slightly different for this release if earlier versions of MAXAda exist on the target machine.

The **`-a check`** option to **`pkgadd`** (see **`pkgadd(1)`**) must be used in this case. The user will see messages similar to the following during the installation. Answer the questions according to the following example:

```
The following files are already installed on the system and are being used by another
package:
         /usr/ada <attribute change only>
         /usr/ada/bin <attribute change only>
         /usr/ada/bin/a.analyze
         /usr/ada/bin/a.build
         /usr/ada/bin/a.cat
         /usr/ada/bin/a.chmod
         /usr/ada/bin/a.compile
         /usr/ada/bin/a.demangle
         /usr/ada/bin/a.deps
         /usr/ada/bin/a.dumpsrc
         /usr/ada/bin/a.edit
         /usr/ada/bin/a.error
         /usr/ada/bin/a.expel
         /usr/ada/bin/a.fetch
         /usr/ada/bin/a.freeze
         /usr/ada/bin/a.help
         /usr/ada/bin/a.hide
         /usr/ada/bin/a.install
         /usr/ada/bin/a.intro
         /usr/ada/bin/a.invalid
         /usr/ada/bin/a.link
         /usr/ada/bin/a.ls
         /usr/ada/bin/a.lssrc
         /usr/ada/bin/a.man
         /usr/ada/bin/a.map
         /usr/ada/bin/a.mkenv
         /usr/ada/bin/a.monitor
         /usr/ada/bin/a.netdb
         /usr/ada/bin/a.options
         /usr/ada/bin/a.partition
         /usr/ada/bin/a.path
         /usr/ada/bin/a.pclookup
         /usr/ada/bin/a.pp
         /usr/ada/bin/a.release
         /usr/ada/bin/a.report
         /usr/ada/bin/a.resolve
         /usr/ada/bin/a.rmenv
         /usr/ada/bin/a.rmsrc
         /usr/ada/bin/a.rtm
         /usr/ada/bin/a.slinker
         /usr/ada/bin/a.syntax
         /usr/ada/bin/a.touch
         /usr/ada/bin/a.trace
         /usr/ada/bin/a.tweak
         /usr/ada/bin/analyze
         /usr/ada/bin/report
         /usr/ada/sup <attribute change only>
         /usr/lib <attribute change only>
         /usr/lib/HyperHelp <attribute change only>
         /usr/lib/HyperHelp/manuals <attribute change only>
         /usr/lib/HyperHelp/manuals/rm.fts <attribute change only>
         /usr/lib/HyperHelp/manuals/rm.hlp <attribute change only>


Do you want to install these conflicting files [y,n,?,q] y

## Checking for setuid/setgid programs.

This package contains scripts which may have a security impact and which will be
executed during the process of installing this package.

Do you want to continue with the installation of this package [y,n,?] y
```

5.0. Overview of MAXAda 3.3.3

MAXAda 3.3.3 is a maintenance release containing all patches and development updates to MAXAda 3.3.2 since its release. See “Changes in This Release” on page 9 for a detailed listing of those changes.

MAXAda 3.3.3 is based on MAXAda 3.1 which was certified using Version 2.1 of the Ada Conformity Assessment Test Suite (certificate #A981215E2.1-047).

MAXAda 3.3.3 supports the Ada95 standard, ANSI/ISO/IEC-8652:1995 as indicated in the following table:

Sections 1 - 13	SUPPORTED
Annex A - Predefined Language Environment	SUPPORTED
Annex B - Interfaces to Other Languages	SUPPORTED
Annex C - Systems Programming	SUPPORTED (*with exceptions**)
Annex D - Real-Time Systems	SUPPORTED (*with exceptions**)
Annex E - Distributed Systems	NOT SUPPORTED
Annex F - Information Systems	NOT SUPPORTED
Annex G - Numerics	NOT SUPPORTED
Annex H - Safety and Security	NOT SUPPORTED
Annex J - Obsolescent Features	SUPPORTED

* The following features are not supported by this implementation:

Feature	RM Reference
Recommended representation support for the following clauses: 13.1(22) - support of non-static constant expressions 13.3(19) - inhibit optimizations based on assumptions of no aliases 13.3(35) - page alignment of standalone library-level objects	C.2
Preelaboration requirements	C.4
Atomic objects are not always moved indivisibly	C.6(15)
Not all storage associated with attributes of a task is reclaimed upon task termination	C.7.2(17)
`Ada.Asynchronous_Task_Control` package not provided or supported	D.11

Details regarding support for Annex C, Annex D, and all implementation-dependent portions of the language can be found in Appendix M of the *MAXAda Reference Manual* (0890516).

5.1. Cross-Compilation

Generally, when compiling natively, programs built on a specific system type and operating system revision will only run correctly on systems of the same architecture and operating system revision. (Some programs which are restricted to generic Ada code may work on multiple system types, but this is not guaranteed.)

However, when cross-compiling, the appropriate **`-osversion`** and **`-arch`** link options must be specified. See the section titled "Link Options" in the "MAXAda Utilities" chapter of the *MAXAda Reference Manual* for information on using these options. Cross-compiling on a PowerMAX system generally requires installation of the PowerMAX Cross-Development Packages (see the *PowerMAX Cross-Development Packages Release Notes* (0891088) for further information).

5.2. System-specific Features

The architectures supported by MAXAda have a variety of real-time features. Not all real-time features are supported by all architectures. As a result, not all MAXAda features are supported by all architectures, and some MAXAda features behave slightly differently on different architectures, because of alternate implementations.

The following MAXAda feature is unavailable on Motorola single board computers, PowerStack II and PowerStack III, Power Hawk 610, 620, and 640 systems, the Power Hawk 700 Series, and the Power Hawk 900 series:

- Pragma `MEMORY_POOL` specifying `LOCAL` memory pools.

The following feature is unavailable on uniprocessor Motorola single board computers, uniprocessor PowerStack II and PowerStack III, Power Hawk 610 and 620 systems, and the Power Hawk 710:

- Pragma `TASK_CPU_BIAS` specifying multiple or nonexistent CPUs.

The following features operate slightly differently on the Motorola single board computer, PowerStack II and PowerStack III, Power Hawk 610, 620, and 640 systems, the Power Hawk 700 Series, and the Power Hawk 900 Series. Consult the reference manual for more information.

- Package `Interval_Timer`
- Package `User_Level_Interrupts`

The following features are unavailable on Motorola single board computers, PowerStack II and PowerStack III, Power Hawk 610, 620 and 640 systems, the Power Hawk 700 series, and the Power Hawk 900 series which lack RCIM boards:

- Package `Ada.Interrupts.ETI_Control`
- Package `ETI_Services`
- Constants `Ada.Interrupts.Names.eti`*nn*

The following features are unavailable on any systems which lack RCIM boards:

- Package `Ada.Interrupts.Distrib_Control`
- Package `Distrib_Services`
- Constants `Ada.Interrupts.Names.distrib`*n*
- Package `Ada.Interrupts.Pig_Control`

5.3. Changes in This Release

The following changes were made in MAXAda Version 3.3.3:

- The implementation of `ada.calendar.time` was changed to support systems with increased processor and clock speeds.

  The size of `ada.calendar.time` has increased from 64 bits to 96 bits.

  It is possible that user-defined records include components of type `ada.calendar.time` and representation clauses may need to be adjusted to account for the change in size.

- MAXAda 3.3.3 improves upon previous MAXAda releases with respect to using subtype range constraints in order to avoid constraint checks. Consider the following example:

  ```
  subtype S is integer range 1..100;
  obj : array (S) of integer;
  index : S := ...;
  ...
  ... obj(index) ...
  ... obj(index) ...
  ... obj(index) ...
  ...
  ```

  The only constraint check will be during the assignment to `index`, if the subtype of the expression has different constraints than `S`. The use of `index` in subsequent array indexing operation will avoid the unnecessary constraint checks.

All items listed below were fixed in patches or development updates to MAXAda Version 3.3.2 and are included in this release:

- The function `ada.exceptions.addresses.propagation_map` was returning an array slice with bounds that were inappropriate for the array type.
- In rare cases, the compiler could create two definitions of the same linker symbol: one in the object file for the specification of a unit and the other in the object file for its body. When linking, an error of the following form would be issued:

  ```
  ld: ...: fatal error: symbol `A$dw_UNIT__...` multiply-defined,
      also in file ...
  ```

  This problem could happen only when using the **`-g`** compilation option. One circumstance in which this occurred was when the specification and body each had as its first declaration a function renaming which denoted an implicit operator.

- A runtime routine associated with tagged type class membership was inappropriately traversing a linked list which could cause segmentation faults under extremely rare circumstances (usually associated with systems whose physical memory is corrupted by inappropriate DMA operations or some other pathological condition).
- Tasks marked with pragma `fast_interrupt_task` and protected object interrupt couriers inappropriately attempted `priocntl` system calls at interrupt level which resulted in additional unnecessary overhead.
- Tasks marked with pragma `fast_interrupt_task` would incorrectly take the "else" path of a conditional entry call to other tasks even if the entry call succeeded.
- A `'Size` attribute applied to a volatile object whose size was dynamic and potentially not an integral number of bytes would fail with an internal assertion error.

- In subprograms with at least a few moderately large composite objects, and then a very large number (typically greater than 1000) scalar or composite objects, a heuristic designed to make the best use of the PowerPC stack frame would fail and produce the error:

  ```
  RM M: declarations require too much space
  ```

  A new option has been added to control the heuristic:

  `-Qframe_scalar_reserved=`*p* (where *p* is a percentage)

  The value can be increased from the default value of 20 if the above error is encountered.

- The **`ar`** command, used when linking archive partitions, had a dependency on the C++ 5.2 release. That dependency has been removed.
- The compiler would fail with an internal error if it encountered an erroneous component selection with a function call prefix and a nonexistent component, and the function call prefix was of an overloaded function, and one of the other functions with the same defining name returned a scalar type.
- The compiler would fail occasionally with an internal error when compiling generic instance bodies which included complex expressions involving implicit conversions of named numbers defined in other generic instances.
- Applications which handled machine interrupts could occassionally cause system panics due to pages faults at interrupt-level on systems with low-memory conditions.
- `Ada.Text_IO.Float_IO.Get` was incorrectly parsing floating point numbers of the form 3.14159E01-5.4.
- Corrects problems with `Ada.Text_IO.Float_IO.Get` improperly rejecting alternative syntactic forms of floating point values.
- Corrects a rare code generation register allocation problem with global optimization (**`-O3`**) involving `OUT` arguments.
- Enhances MAXAda's `DISTRIBUTED_LOCAL_LOCKING` pragma to allow for an option argument, `TRUE` or `FALSE`. The semantics for a value of `TRUE`, which is equivalent to use of the pragma without an argument, remains unchanged and instructs the runtime system to ensure that the process is "distributed" during elaboration of the program so that subsequent migration requests to other CPUs can be granted in the case of Non Uniform Memory Architecture (NUMA) systems where the application uses memory pages which are locked in local memory. Supplying a value of `FALSE` will prevent the runtime system from "distributing" the process during elaboration. The pragma was provided to allow the user to restrict the runtime from worst-case assumptions when it detected use of variable CPU biases, interrupt handlers, etc. By specifying a value of `FALSE`, the user effectively declares that the application does not require use of CPUs from multiple CPU boards or such use does not involved locked local memory pages.

WARNING

Users which have existing configuration files from **`a.map`** which are used to modify an application's behavior must obtain new map files (by rerunning **`a.map`**) because of an incompatibility in the `DISTRIBUTED_LOCAL_LOCKING` setting. When the pragma is not used, the setting now defaults to "DEFAULT". Previous map files described the default setting as "FALSE", which now has the new semantics described above.

- A `'Model` attribute with a static value would fail with an internal error.
- A `'Machine` attribute with a static value did not round the value to a machine number.
- A static expression involving `'Leading_Part` where the Radix parameter is 0.0 was not detected as an error at compile time.
- Constraint checks associated with conversions or allocators of composite types with discriminants of access-to-array types could fail with internal compiler errors.

5.4. Known Deficiencies

5.4.1. Optimization

Use of maximal optimization (**`-O3`**) has been heavily tested, but not as rigorously as the default level, `MINIMAL` (**`-O1`**).

Optimization levels above **`-O1`** should be reserved for fully tested programs.

Language features that are known to be problematic include:

- controlled types
- `'Valid` attribute

5.4.2. Atomic Components

The compiler currently does not ensure that atomic components of composite types are copied with indivisible read and write operations when their containing composite objects are copied as a whole.

5.4.3. Nested Asynchronous Select Statements

Asynchronous select statements nested within other asynchronous select statements are known to fail occasionally in this release. This includes cases where the nesting is not obvious because of subprogram calls. The failures occur when the triggering statement in the outer asynchronous select statement completes before the outer abortable part completes and before the inner triggering statement completes.

5.4.4. Max_Size_In_Storage_Elements Attribute

The `Max_Size_In_Storage_Elements` attribute may evaluate to an incorrect result or raise `Constraint_Error` if its value would be outside the range of `Standard.Integer`.

5.4.5. Incomplete Types

If a library package declaration contains an incomplete type which must be completed in its body, and the body is not required for any other reason (e.g. no subprogram declarations, nor pragma `Elaborate_Body`), **`a.build`** may issue a warning initially (and after any change to the source file containing the library package declaration) that the body is not required and will not be included in partitions that require the library package. The warning is false, and is corrected automatically by the compiler before the partitions are linked. It should be ignored.

6.0. Direct Software Support

Software support is available from a central source. If you need assistance or information about your system, please contact the Concurrent Software Support Center at 1-800-245-6453. Our customers outside the continental United States can contact us directly at 1-954-283-1822 or 1-305-931-2408. The Software Support Center operates Monday through Friday from 8 a.m. to 7 p.m., Eastern Standard time.

Calling the Software Support Center gives you immediate access to a broad range of skilled personnel and guarantees you a prompt response from the person most qualified to assist you. If you have a question requiring on-site assistance or consultation, the Software Support Center staff will arrange for a field analyst to return your call and schedule a visit.