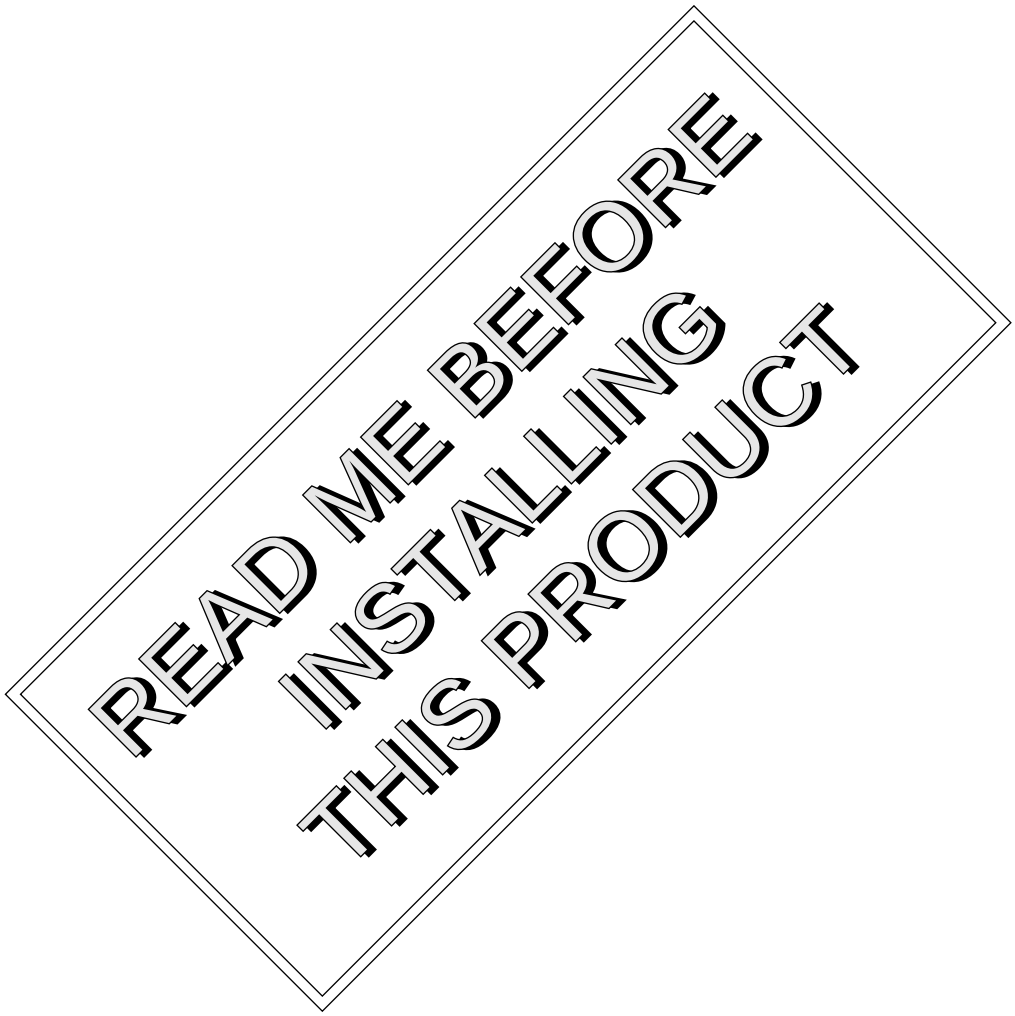
MAXAda

Version 3.3.3 Release Notes (Linux)

August 2003

0898516-3.3.3



CONCURRENT
COMPUTER
CORPORATION

Copyright

Disclaimer

The information contained in this document is subject to change without notice. Concurrent Computer Corporation has taken efforts to remove errors from this document, however, Concurrent Computer Corporation's only liability regarding errors that may still exist is to correct said errors upon their being made known to Concurrent Computer Corporation.

License

Trademark Acknowledgments

Contents

1.0. Introduction

MAXAda™ is part of the PowerWorks™ Linux Development Environment (PLDE) and supports development of Ada95 programs running under Concurrent Computer Corporation's PowerMAX OS™. 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 a maintenance release containing all patches and development updates to MAXAda 3.3.2 since its release. See "Changes in This Release" on page 11 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).

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.

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 (Linux)	0898516-3.3.3

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 PowerWorks Linux Development Environment utility, **`nhelp`**
- in PDF format in the **`documentation`** directory of the PLDE Installation CD
- on the Concurrent Computer Corporation web site at www.ccur.com

3.0. Prerequisites

Prerequisites for MAXAda Version 3.3.3 for both the host system and target system are as follows:

3.1. Host System

3.1.1. Software

- RedHawk™ Linux *or* Red Hat® Linux*
- Required capabilities
  - PowerWorks Linux Development Environment

NOTE

The following capabilities are normally installed as part of the standard installation of the PowerWorks Linux Development Environment. During installation of the PLDE, the user will be notified if required capabilities do not exist on the Linux system.

Capabilities	**RPMs providing these capabilities**
plde-HyperHelp **plde-HyperHelp-scripts** **plde-nhelp** **plde-pmax-crossdev**	**plde-x11progs-6.4.2-006** **plde-HyperHelp-scripts-6.4.2-002** *any or all of the following:* **plde-pmax-crossdev-4.3-P12-1** **plde-pmax-crossdev-5.1-SR6-1**

User applications built with MAXAda may require other capabilities which are provided by additional RPMs included on the PLDE Installation CD. Refer to the section titled "Cross-Development Libraries and Headers" in the *PowerWorks Linux Development Environment Release Notes* (0898000) for more information.

3.1.2. Hardware

- an Intel®-based PC - 300Mhz or higher (recommended minimum configuration)
- 64MB physical memory (recommended minimum configuration)

* This product has been extensively tested on RedHawk Linux 1.3 and Red Hat Linux 7.3 and 8.0. However, this product has not been tested with versions of Linux supplied by other vendors or on earlier versions of Red Hat Linux.

3.2. Target System

3.2.1. Software

- PowerMAX OS 4.3 or later

3.2.2. 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

4.0. System Installation

Installation of the host portion of MAXAda is normally done as part of the general installation of the PowerWorks Linux Development Environment software suite. A single command installs (or uninstalls) all software components of the PLDE, as described in the *PowerWorks Linux Development Environment Release Notes* (0898000).

The following section describes how to install (or uninstall) MAXAda separately from the PLDE suite for those rare cases when this is necessary.

In addition, it is necessary to install the PowerMAX OS cross-development libraries on your Linux system in order to cross-link on that system. See "Cross-Development Libraries" on page 6 for more information.

Further, it is necessary to activate the MAXAda product prior to use. See "Activation" on page 7 for more information.

4.1. Separate Host Installation

In rare cases, it may be necessary to install (or uninstall) MAXAda independent of the installation of the PowerWorks Linux Development Environment software suite. This may be done using the standard Linux product installation mechanism, **`rpm`** (see **`rpm(8)`**).

The names of the RPMs associated with MAXAda 3.3.3 are:

```
plde-MAXAda-3.3.3
plde-MAXAda-invoker
plde-MAXAda-rm
```

and the files associated with these RPMs, respectively, are:

```
plde-MAXAda-3.3.3-000-1.i386.rpm
plde-MAXAda-invoker-3.3.3-000.i386.rpm
plde-MAXAda-rm-3.3.3-000.i386.rpm
```

which can be found in the **`linux-i386`** directory on the PowerWorks Linux Development Environment Installation CD.

NOTE

The user must be root in order to use the **`rpm`** product installation mechanism on the Linux system.

To install the MAXAda RPMs, issue the following commands on your Linux system:

1. Insert the PowerWorks Linux Development Environment Installation CD in the CD-ROM drive
2. Mount the CD-ROM drive (assuming the standard mount entry for the CD-ROM device exists in **`/etc/fstab`**)

   ```
   mount /mnt/cdrom
   ```

3. Change the current working directory to the directory containing the MAXAda RPMs

   ```
   cd /mnt/cdrom/linux-i386
   ```

4. Install the RPMs

   ```
   rpm -Uvh plde-MAXAda-invoker-3.3.3-000.i386.rpm \
            plde-MAXAda-3.3.3-000-1.i386.rpm \
            plde-MAXAda-rm-3.3.3-000.i386.rpm
   ```

   By default, the product is installed in **/usr/opt**.

5. Change the current working directory outside the **/mnt/cdrom** hierarchy

   ```
   cd /
   ```

6. Unmount the CD-ROM drive (otherwise, you will be unable to remove the PowerWorks Linux Development Environment Installation CD from the CD-ROM drive)

   ```
   umount /mnt/cdrom
   ```

To uninstall the MAXAda RPMs, use the following command:

```
rpm -e plde-MAXAda-rm \
       plde-MAXAda-invoker \
       plde-MAXAda-3.3.3
```

4.2. Cross-Development Libraries

In order to cross-link on the Linux system, it is necessary to have certain PowerMAX OS libraries installed on that system.

The names of the RPMs containing the PowerMAX OS libraries minimally required for cross-linking are:

```
plde-pmax-crossdev-4.3
plde-pmax-crossdev-5.1
```

and are used when linking for a PowerMAX OS 4.3 and PowerMAX OS 5.1 target system, respectively. The files associated with these RPMs are:

```
plde-pmax-crossdev-4.3-P12-1.i386.rpm
plde-pmax-crossdev-5.1-SR6-1.i386.rpm
```

NOTE

The version number is part of the name of the RPM. Because of that, it is possible to install both RPMs on the Linux system at the same time. This allows the user to generate executables for multiple PowerMAX OS versions from the same Linux system.

User applications built with MAXAda may require other capabilities which are provided by additional RPMs included on the PLDE Installation CD. Refer to the section titled “Cross-Development Libraries and Headers” in the *PowerWorks Linux Development Environment Release Notes* (0898000) for more information.

4.3. Activation

The MAXAda product requires activation before it can be utilized. Installation of MAXAda automatically activates the product for a 30-day evaluation period. During that time period, you can utilize the product for purposes of evaluation only. Permanent activation requires an activation code. Contact Concurrent Software Distribution by calling 1-800-666-5405 or by faxing 1-800-666-5404. Our customers outside the continental United States can contact Concurrent Software Distribution by calling 1-954-283-1836 or by faxing 1-954-283-1835. Concurrent Software Distribution may also be reached by email at softdist@ccur.com. If you purchased MAXAda, the activation code is provided on the Installation CD case.

Issue the following command as the root user to permanently activate the product on your system:

`/usr/ada/bin/a.install -rel phase3.3.3 -activate` *code*

where *code* is the code provided on the Installation CD case.

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 11 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. Cautions

6.1. PowerMAX OS and Linux Compatibility

The MAXAda component of the PowerWorks Linux Development Environment is a cross-compilation system. The object files, archives, and shared libraries produced by it are not compatible with Linux and will not run in that environment. In general, Linux binary utilities (e.g. **`binutils`**) will be unable to interpret them.

Object files, archives, and shared libraries produced by MAXAda are fully compatible with similar objects produced on PowerMAX OS systems.

However, the compilation environments themselves cannot be shared between PowerMAX OS and Linux systems, due to their internal representations (byte ordering).

For example, if two environments were created using PowerMAX OS and Linux versions of MAXAda, attempts to share the environments would result in an error similar to that shown below:

PowerMAX OS system:

```
mkdir /pmax_disk/env
cd /pmax_disk/env
a.mkenv -rel phase3.3.3
```

Linux system:

```
mkdir /linux_disk/env
cd /linux_disk/env
a.mkenv -rel phase3.3.3 -osversion osversion -arch arch
a.path -i /pmax_disk/env
a.ls -all
a.ls: fatal: environment /pmax_disk/env was built with endianness
BIG, but this tool expects endianness LITTLE: environment built
with incompatible host architecture.
```

6.2. Operational Differences between PowerMAX OS and Linux

MAXAda operates almost identically on a Linux system as it does on a PowerMAX OS system with the following exceptions:

- to link programs or shared objects on Linux, the **`-arch`** and **`-osversion`** link options must be supplied, either on a per-partition basis or associated with the environment

  See the sections titled “OS Version” and “Target Architecture” under the “Link Options” section of the “MAXAda Utilities” chapter of the *MAXAda Reference Manual* (0890516) for more information.

- **`a.map`** is not available on the Linux system but may be executed on the PowerMAX OS system
- **`a.monitor`** is not available on the Linux system but may be executed on the PowerMAX OS system
- **`a.rtm`** is not available on the Linux system but may be exectued on the PowerMAX OS system

- **a.analyze** and **a.report** are not currently available on the Linux system, but may be utilized on the target system
- selective linking (the **-c** link option) is not available

The **a.map**, **a.monitor**, and **a.rtm** tools are provided on the PowerWorks Linux Development Environment Installation CD in the **powermax-ppc604/MAXAda** directory. They may be copied from that location and placed on a PowerMAX OS system for execution. The **a.analyze** and **a.report** utilities are not available on the PLDE Installation CD; however, their generic PowerMAX OS counterparts, **analyze** and **report**, are available on all PowerMAX OS systems.

6.3. TASK_UNMAPPED_BASE

MAXAda has a requirement that the Linux kernel not be rebuilt with a TASK_UNMAPPED_BASE value of less than 0x40000000 (1GB). That could happen because of a change to any of the following macros in the kernel:

/usr/src/linux*/include/linux/autoconf.h:

```
#define CONFIG_1GB 1
```

/usr/src/linux*/include/asm*/page_offset.h:

```
#define PAGE_OFFSET_RAW 0xC0000000
```

/usr/src/linux*/include/asm*/page.h:

```
#define __PAGE_OFFSET          (PAGE_OFFSET_RAW)
#define PAGE_OFFSET            ((unsigned long)__PAGE_O
```

/usr/src/linux*/include/asm*/processor.h:

```
#define TASK_SIZE              (PAGE_OFFSET)
#define TASK_UNMAPPED_BASE     (TASK_SIZE / 3)
```

7.0. 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.