MAXAda for Linux

Version 3.5.1-SR3 Release Notes

May 2015

0898537-3.5.1-SR3

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

1.0. Introduction

MAXAda™ for Linux® supports development of Ada95 programs running under selected Linux distributions. MAXAda for Linux 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 for Linux 3.5.1 is based on MAXAda 3.1 which was certified using Version 2.1 of the Ada Conformity Assessment Test Suite (certificate #A981215E2.1-047).

Refer to “Overview of MAXAda for Linux” on page 16 for more information on implementation dependent features.

In addition, MAXAda for Linux 3.5.1 includes POSIX® 1003.5, a complete implementation of the Institute of Electrical and Electronic Engineers (IEEE) standard IEEE-Std-1003.5-1992.

This service release, MAXAda for Linux 3.5.1-SR3, includes all MAXAda updates since the original release of 3.5.1. Many of these updates are required to be able to install MAXAda, build MAXAda programs, and execute them on Linux versions that were not available at the time of the original release of 3.5.1; RedHawk 7.0 being the most recent example.

The remainder of this document remains mostly unchanged from the original release. Modifications to this document for Version 3.5.1-SR3 are marked with “change bars”, as is this paragraph.

See “New Features in MAXAda for Linux 3.5.1-SR3” on page 16 for more information on new features included in this service release.

2.0. Documentation

Table 2-1 lists the MAXAda for Linux 3.5.1-SR3 documentation available from Concurrent.

Table 2-1. MAXAda for Linux Version 3.5.1-SR3 Documentation

Manual Name	Pub. Number
MAXAda for Linux Reference Manual	0898537-140
MAXAda for Linux Version 3.5.1-SR3 Release Notes	0898537-3.5.1-SR3

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.

Additionally, the manuals listed above are available:

- after installation using the utility **`/usr/ada/bin/a.man`**
- in PDF format in the **`documentation`** directory of the *MAXAda for Linux 3.5.1-SR3 Installation CD*
- on the Concurrent Computer Corporation web site at **`http://redhawk.ccur.com/docs.`**

3.0. Prerequisites

Prerequisites for MAXAda for Linux Version 3.5.1-SR3 for both the host system and target system are as follows:

3.1. Host System

3.1.1. Software

Any of the following operating systems:

- RedHawk Linux Version 2.1 or later
- Red Hat Enterprise Linux Versions 3 through 7
- CentOS Versions 5 through 7

3.1.2. Hardware

- Any i386 or x86_64 system supported by the host operating system

3.2. Target System

MAXAda supports self-hosted targets as well as cross compilation between compatible systems.

When cross-compiling for target systems, the host system operating system type and version must match the underlying target system operating type and version. For example, a system running Red Hat Enterprise Linux 3.0 is an appropriate host for a RedHawk Linux 2.3 target system, whereas a system running CentOS 7 or RHEL 7 is an appropriate host for a target system running RedHawk Linux 7.0.

When cross-linking, any required libraries specific to the target system must be present on (or accessible from) the host system.

For example:

```
> mkdir -p /redhawk-root/usr/lib
> scp redhawk:/usr/lib/libccur_rt.* /redhawk-root/usr/lib
> a.partition -default -oappend -L/redhawk-root/usr/lib
```

The former two commands setup and populate a local directory with a library from the target system.

The latter command sets the default link options for the current environment (assuming the MAXAda environment already exists) to search **`/redhawk-root/usr/lib`** when searching for libraries.

3.2.1. Software

- RedHawk Linux 2.1 through 7.0

3.2.2. Hardware

- Any i386 or x86_64 system supported by RedHawk

4.0. Installation

The *MAXAda for Linux Installation CD* contains all RPMs specifically required for operation of the MAXAda compiler, linker, and utilities.

The following table shows the RPMs that will be installed.

Item	RPM
MAXAda	*32-bit Systems:* **ccur-MAXAda-suite** **ccur-MAXAda-invoker** **ccur-MAXAda-rm** **ccur-MAXAda-i86_3.5.1** *64-bit Systems:* **ccur-MAXAda-suite** **ccur-MAXAda-invoker** **ccur-MAXAda-rm** **ccur-MAXAda-amd64_3.5.1**
NightBench	**ccur-nbench-2.3.2** **ccur-nbench-ada-2.3.2**
Scripts	**ccur-HyperHelp-scripts**
Utilities	**ccur-x11progs**

4.1. Standard Installation

To install MAXAda for Linux 3.5.1-SR3, follow the steps below on your Linux system as the `root` user:

1. Insert the *MAXAda for Linux 3.5.1-SR3 Installation* disk in the CD/DVD drive. On newer systems it will automatically mount as one of the following:

   /media/MAXAda-3.5.1-SR3
   /run/media/`root`**/MAXAda-3.5.1-SR3**

2. If the disk does not auto-mount, mount the disk drive manually using commands similar to the following, substituting your actual disk drive device as required;

   ```
   mkdir /tmp/mnt
   mount -t iso9660 -o ro,exec /dev/sr0 /tmp/mnt
   ```

3. Change the current working directory to the now-mounted disk base directory, whatever that may be (see steps 1-2 above).

   cd /run/media/`root`**/MAXAda3.5.1-SR3**

4. Invoke the MAXAda for Linux installation script

   ./install-maxada

5. Change the current working directory outside of the mounted disk hierarchy

   cd /

6. Eject the disk or manually unmount it if you had to manually mount it in step 2

   ```
   eject
     - or -
   umount /tmp/mnt
   ```

7. To remove the MAXAda for Linux RPMs, follow the steps above to mount the CD and issue the following command:

   ./uninstall-maxada

4.2. Installing MAXAda 32-bit on a 64-bit system

Starting with MAXAda Version 3.5.1-SR3 and RedHawk 7.0, MAXAda supports the creation and execution of 32-bit programs on 64-bit systems.

Since MAXAda allows multiple versions to be installed on the same system, you can install the 32-bit version directly on a 64-bit system.

In reality, this can be done on Linux distributions prior to RedHawk 7.0; however, typically such distributions only had partial 32-bit support. Starting with RedHawk 7.0 (CentOS7/RHEL7), 32-bit development and execution is fully* supported on 64-bit systems.

After the installation of the 64-bit version, navigate to the **RPM/i386** directory of the mounted installation disk and issue the following commands:

```
rpm -Uvh ccur-MAXAda-i86_3.5.1*.i386.rpm
```

The following command lists the releases that are available on the system. In this case the word release identifies both the version number and the architecture type of MAXAda installation. Issue the following command:

```
/usr/ada/bin/a.release
```

You will see output similar to the following:

```
The following releases are available on maxada_host:
 Name            Path
 ----            ----
 amd64_3.5.1     /usr/ada/amd64_3.5.1
*i386_3.5.1      /usr/ada/i86_3.5.1

(*) Designates the system default release

The predefined release installation, "default", is also available.
```

The last release installed will be the default; in the case above, that is i386_3.5.1. Most likely your system administrator will want the self-hosted version marked as the default; issue the following command to do that:

```
/usr/ada/bin/a.install -rel amd64_3.5.1 -d
```

When you create a MAXAda build environment, you can select the release you wish to use. The following command creates at 32-bit MAXAda development environment:

/usr/ada/bin/a.mkenv -rel i86_3.5.1 *dirname*

The above creates the directory *dirname* and sets its release to i86_3.5.1 (the "release" name of the 32-bit version of MAXAda 3.5.1). All subsequent operations within that environment will utilize that release.

You can query the release version for any MAXAda environment using the command

```
/usr/ada/bin/a.release -q
```

* CentOS 7 and RHEL 7 generally provide access to all common 32-bit libraries needed for linking or execution. The user is responsible for installing any required 32-bit RPMs that are not installed by default.

You can set the (persistent) default release for a specific user using the following command, while logged in as the user of interest:

`/usr/ada/bin/a.release -rel` *relname* **`-u`**

where *relname* is either i86_3.5.1 or amd64_3.5.1. See **`a.release`** and **`a.install`** in the *MAXAda Reference Manual* for more information.

4.3. System Configuration

RedHawk Linux comes with default system settings which are sufficient for most Ada programs. If you intend to compile and build Ada programs on systems without the RedHawk kernel, you **must** address "Randomization of Address Space" on page 8.

Changes to system control variables may benefit or impair some Ada programs. Consider the following issues:

- Randomization of Address Space
- Capabilities
- Shared Memory Limits

See **sysctl(8)** for more information on kernel configuration parameters.

4.3.1. Randomization of Address Space

Use of the MAXAda compilation system requires that address space randomization is turned off.

To determine this, invoke the following command:

```
/usr/sbin/sysctl -a | fgrep kernel_randomize_va_space
```

If the variable is not set to zero, issue the following command:

```
/usr/sbin/sysctl -w kernel_randomize_va_space=0
```

to have it take effect immediately. Additionally, add the command to **/etc/rc.d/rc.local** boot script to ensure it is zero on subsequent reboots.

NOTE:

Older systems may not have the randomization feature.

NOTE:

This system control setting is not required to be turned off to execute MAXAda programs, only to build programs with MAXAda.

4.3.2. Capabilities

Linux provides a means to grant otherwise unprivileged users the authority to perform certain privileged operations. **pam_capability(8)**, the Pluggable Authentication Module, is used to manage sets of capabilities, called roles, required for various activities.

Linux systems should be configured with an `adauser` role which provides the capabilities required by MAXAda for Linux. In order to run MAXAda tasking programs on a target, each MAXAda for Linux user must be configured to use (at a minimum) the capabilities specified below.

Edit **/etc/security/capability.conf** and define the `adauser` role (if it is not already defined) in the "ROLES" section:

```
role adauser cap_sys_admin cap_sys_nice cap_sys_rawio cap_ipc_lock
```

Additionally, for each MAXAda for Linux user on the target system, add the following line at the end of the file:

user *username* adauser

where *username* is the login name of the user.

If the user requires capabilities not defined in the `adauser` role, add a new role which contains `adauser` and the additional capabilities needed, and substitute the new role name for `adauser` in the text above.

In addition to registering your login name in **`/etc/security/capability.conf`**, files under the **`/etc/pam.d`** directory must also be configured to allow capabilities to be activated.

To activate capabilities, add the following line to the end of selected files in **`/etc/pam.d`** if it is not already present:

```
session required pam_capability.so
```

The list of files to modify is dependent on the list of methods that will be used to access the system. The following table presents a recommended configuration that will grant capabilities to users of the services most commonly employed in accessing a system.

/etc/pam.d File	Affected Services	Comment
`remote`	telnet rlogin rsh (when used w/o a command)	Depending on your system, the **`remote`** file may not exist. Do not create the **`remote`** file, but edit it if it is present.
`password-auth`	Most all login mechanisms.	This file is present in recent OS distributions. If it is present add the clause mentioned above to this file.
`login`	local login (e.g. console) telnet* rlogin* rsh* (when used w/o a command)	*On some versions of Linux, the presence of the **`remote`** file limits the scope of the **`login`** file to local logins. In such cases, the other services listed here with **`login`** are then affected solely by the **`remote`** configuration file.
`rsh`	rsh (when used with a command)	e.g. **`rsh system_name a.out`**
`sshd`	ssh	You must also edit **`/etc/sshd/sshd_config`** and ensure that the following line is present: **`UsePrivilegeSeparation no`**
`gdm`	gnome sessions	
`kde`	kde sessions	

If you modify **`/etc/pam.d/sshd`** or **`/etc/sshd/sshd_config`**, you must resetart the **`sshd`** service for the changes to take effect:

On RedHawk systems: **`service sshd restart`**

In order for the above changes to take effect, the user must log off and log back onto the target system.

NOTE

To verify that you have been granted capabilities, issue the following command:

```
/usr/sbin/getpcaps $$
```

The output from that command will list the roles currently assigned to you.

See the section titled “Capabilities” in the “Introduction to MAXAda” chapter of the *MAXAda Reference Manual* (0898537) for more information.

4.3.3. Shared Memory Limits

The following kernel configuration parameters may need to be increased for Ada programs using large numbers of shared packages (Pragma shared_package) or shared packages with large size requirements.

- *kernel.shmmni*
- *kernel.shmall*
- *kernel.shmmax*

5.0. Getting Started

MAXAda provides a command-line interface as well as graphical interface to the compilation and link processes.

The graphical interface is provided by a legacy tool called NightBench (**`nbench`**). While this tool has an old look and feel (written in Motif) and lacks many features one would find in a modern IDE, it can be very useful, especially for new users. See the "Using NightBench with Ada" section in the *NightBench User's Guide* (0890514) (**`/usr/ada/bin/a.man nbench`**) for more information.

A brief introduction to the command line interface follows in the next two sub-sections. You should refer to "Using MAXAda" section in the *MAXAda Reference Manual* (0890516) (**`/usr/ada/bin/a.man maxada`**) for a much more detailed description.

To utilize the command-line interface, add the following to your `PATH`:

`/usr/ada/bin`

5.1. MAXAda Hello World

The following steps show how to build and execute a “Hello World” program.

- Ensure your PATH variable includes the MAXAda utility directory

```
PATH=$PATH:/usr/ada/bin
export PATH
```

NOTE

> Do no put the release-specific path (as seen in **a.release** output) in your PATH variable. MAXAda utilities automatically figure out the correct release for the current environment or the environment specified with the **-rel** option, which can be specified with any MAXAda command.

- Create a source file in the environment called **hello.ada** as shown here

```
cat << EOF > hello.ada
with ada.text_io;
procedure hello_world is
begin
   ada.text_io.put_line(“MAXAda says hello”);
end hello_world;
EOF
```

- Enter the following commands

```
a.mkenv -g
a.intro hello.ada
a.partition -create active hello_world
a.build -v hello_world
```

The **a.mkenv** command creates a MAXAda build environment in the current working directory. It is easiest to think of the directory and the build environment within it as a single entity although the environment actually consists of a **.Ada** file and a **.ada** subdirectory within the directory.

The **a.intro** command introduces new Ada source files to the environment. They must be generally syntactically correct for the command to succeed.

The **a.partition** command defines the Ada program to be built (an *active* partition is Ada-speak for an executable program; other partition types, such as *library*, can be defined).

The a**.build** command is the workhorse. You typically will be invoking **a.build** (and possibly **a.compil**e and **a.link** which **a.build** invokes itself) most of the time. As shown above, it will compile and link the Ada program, resulting in an executable file called **hello_world**.

- Execute the program just built

```
./hello_world
```

- If you needed to debug the program, you would invoke it with NightView* as shown here:

```
nview ./hello_world
```

You can obtain a short description of any MAXAda utility by invoking the utility with a **-H** option. You can obtain full information about a utility by invoking **a.man** with the utility name as an argument. For example:

a.man *a.build*

Refer to "Using MAXAda" section in the *MAXAda Reference Manual* (0890516) (or **/usr/ada/bin/a.man maxada**) for much more information.

* NightView is a symbolic real-time debugger that is part of the NightStar RT product; NightView highly recommended for use with MAXAda.

6.0. Compliance with ANSI/ISO/IEC-8652:1995

MAXAda for Linux 3.5.1-SR3 is based on MAXAda 3.1 which was certified using Version 2.1 of the Ada Conformity Assessment Test Suite (certificate #A981215E2.1-047).

6.1. Support Table

MAXAda for Linux 3.5.1-SR3 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
Pre-elaboration 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).

6.2. Implementation-Dependent Issues

6.2.1. Priorities

The Ada95 language defines priorities in terms of the discrete subtypes defined in the package `System`. The subtype `any_priority` spans the entire priority range supported by the implementation while the subtypes `priority` and `interrupt_priority` divide that range into standard user-level priorities and interrupt priorities (those which require the blocking of one or more interrupts).

For Ada tasking programs, the default Task Dispatching Policy is `FIFO_Within_Priorities`. The Ada priority values of `system.priority'first` (1) .. `System.priority'past` (98), map directly to the Linux SCHED_FIFO 1..98 priorities.

`System.interrupt_priority'first` (99), maps directly to Linux SCHED_FIFO priority 99.

Use of `System.interrupt_priority'last` (100) is reserved for Protected Actions. All external maskable machine interrupts are masked during such actions. Programs which use this priority value must lock their address space in memory (e.g. `pragma Pool_Lock_State(default,locked)`) and must exercise extreme care inside protected actions. Misuse of this priority value can cause system panics and/or have significant effects on system performance and determinism.

See the section titled "Priorities" in the "Run-Time Concepts" chapter of the *MAXAda Reference Manual* (0890537) for more information on priorities and scheduling classes.

6.2.2. Tasking

MAXAda implements Ada tasking using real-time operating system features which are not available for all users by default. You must grant access to these features for normal users. See "Capabilities" on page 8 for more information.

6.2.3. Bit Numbering

Bit numbering on Pentium and AMD64 systems is different from that on PowerPC systems due to endian differences. These facts must be taken into consideration when interfacing to devices or using `Ada.Unchecked_Conversion` with types that have different signs and sizes;

- `'Bit_Order` is `Low_Order_First`
- `System.Default_Bit_Order` is `Low_Order_First`
- In representation clauses (RM 13.5.1 & 13.5.2), the `first_bit` is the low-order bit and the `last_bit` is the high-order bit.
- In packed arrays, the low-order bit of a component appears in a lower byte number, or in the same byte number but lower bit number, than the component's high-order bit (assuming a component larger than 1 bit).

6.2.4. 'Alignment maximum

On Pentium, `'Alignment` maximum is 4 bytes for stack objects.

On AMD64, `'Alignment` maximum is 16 bytes for stack objects.

7.0. Overview of MAXAda for Linux

This section provides a general overview of MAXAda for Linux 3.5.1 as well as some detailed information about service release 3.5.1-SR3.

7.1. New Features in MAXAda for Linux 3.5.1-SR3

3.5.1-SR3 is largely a maintenance release comprised of all updates to MAXAda 3.5.1 since its original release.

It also includes changes required for using MAXAda and MAXAda-built programs on RedHawk 7.0.

A new feature included in this service release is support for building 32-bit MAXAda programs on a 64-bit system. See “Installing MAXAda 32-bit on a 64-bit system” on page 6 for further information.

7.2. Deprecated Math Package

The deprecated `math` package's handling of error conditions is different on Linux than on PowerMAX OS. For many errors, an exception is raised immediately while performing the math operation instead of merely setting `errno` and postponing the exception until `math.check_errno` is called.

For more information, see the comments in the specification of the `math` package in:

`/usr/ada/`*rel*`/deprecated/math.a`**

where *rel* is either:

`i86_3.5.1` *(Intel)*

or:

`amd64_3.5.1` *(AMD64)*

7.3. Compatibility Issues for RedHawk Systems

MAXAda programs built with version 3.5.1 are generally upward compatible with RedHawk systems starting with RedHawk 2.1 - RedHawk 7.0. Thus programs built with MAXAda 3.5.1 on a RedHawk 5.1 system will generally operate correctly on all RedHawk versions from 5.1 through 7.0.

Backward compatibility is undefined -- generally you should not expect programs built on a more recent operating system to run on an older operating system.

7.3.1. Binary Incompatibility

Due to changes in the Red Hat Enterprise Linux 4 environment, some unusual incompatibilities have arisen, as discussed below in the following two subsections.

MAXAda programs built with version 3.5 for AMD64 systems are incompatible with the RedHawk 4.1 (Red Hat Enterprise Linux 4) C library. Unless your program was statically linked, you will need to rebuilt it using MAXAda 3.5.1. The incompatibility involves any service requiring a `stat` (**`stat(2)`**, **`lstat`**(2), or **`fstat`**(2)) action, which includes all programs which require `ada.text_io`, `posix_files`, or `posix_file_status` or programs which use one of the stat functions defined in `posix_1003_1`.

7.3.2. Source Incompatibility

The RedHawk 4.1 (Red Hat Enterprise 4) libraries have changed the set of reserved signals. This change affects SIGRTMIN, which is no longer a static value.

The MAXAda-supplied package, `posix_1003_1`, previously included a static constant called `SIGRTMIN`. `SIGRTMIN` has been changed to be a function which returns the appropriate value of `SIGRTMIN`, based on the platform and library set in use.

This could cause user declarations to fail to compile if `posix_1003_1.SIGRTMIN` was used as an initial value when a static initial value was required.

In previous releases of Red Hat, SIGRTMIN had the value 32 even though a few signals starting at that value were reserved for use by the threads library.

Depending on your library usage, under RedHawk 4.1, SIGRTMIN may now have a different value and typically all signals starting at SIGRTMIN through SIGRTMAX are now available for general use.

Note:

The MAXAda runtime system continues to reserve signal 47 for its use.

7.4. Legacy Issues

MAXAda for Linux 3.5.1 lacks some features that historically have been available on other MAXAda-supported platforms. These include:

- Position Independent Code - currently MAXAda for Linux only supports statically linked Ada code although MAXAda programs may utilized system shared libraries
- Specification of "restricted interrupt handling" has no effect in this release.
- Pragma `FAST_INTERRUPT_TASK` has no effect in this release.
- **`a.rtm`** is not available in MAXAda for Linux 3.5.1-SR3. The NightProbe tool (**`nprobe`**) is recommended in its place. See the *NightProbe User's Guide* (0890465) for more information.

The following tools are not currently planned for MAXAda for Linux:

- **`a.analyze`**
- **`a.report`**
- **`a.slinker`**

The following packages are not available or are depreciated in MAXAda for Linux:

- `Userdma_Support`

  Linux does not currently support a **`userdma(2)`** service.

- `User_Level_Interrupts`

  Linux does not currently support user-level interrupts.

- `RT_Interface`

  The `RT_Interface` package has been moved to the **`deprecated`** environment. It has been replaced in this release with the package `FBSched` which is available in the **`vendorlib`** environment.

8.0. 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 our toll free number 1-800-245-6453. For calls outside the continental United States, the number is 1-954-283-1822. The Software Support Center operates Monday through Friday from 8 a.m. to 5 p.m., Eastern Standard Time.

You may also submit a request for assistance at any time by using the Concurrent Computer Corporation web site at **http://ccur.com/support** or by sending an email to **support@ccur.com**.

Contents