LynxOS 4.0 Release Notes

LynxOS Release 4.0

DOC 0401-00

Printed in the United States of America.

LynuxWorks, Inc. makes no representations, express or implied, with respect to this documentation or the software it describes, including (with no limitation) any implied warranties of utility or fitness for any particular purpose; all such warranties are expressly disclaimed. Neither LynuxWorks, Inc., nor its distributors, nor its dealers shall be liable for any indirect, incidental, or consequential damages under any circumstances.

(The exclusion of implied warranties may not apply in all cases under some statutes, and thus the above exclusion may not apply. This warranty provides the purchaser with specific legal rights. There may be other purchaser rights which vary from state to state within the United States of America.)

Contents

Preface

For More Information

For more information on the features of LynxOS, refer to the following documentation.

- *LynxOS Installation Guide*

  This manual supports the initial installation and configuration of LynxOS and the X Windows System.

- *LynxOS User's Guide*

  This document contains information about basic system administration and kernel level specifics of LynxOS. It contains a "Quick Starting" chapter and covers a range of topics, including tuning system performance and creating kernel images for embedded applications.

- *Online information*

  The complete LynxOS documentation set is available on the Documentation CD-ROM. Books are provided in both HTML and PDF formats.

  Updates to these documents are available online at the LynuxWorks website: `http://www.lynuxworks.com`.

  Additional information about commands and utilities is provided online with the **`man`** command. For example, to find information about the GNU gcc compiler, use the following syntax:

  ```
  man gcc
  ```

Typographical Conventions

The typefaces used in this manual, summarized below, emphasize important concepts. All references to file names and commands are case sensitive and should be typed accurately.

Kind of Text	**Examples**
Body text; *italicized* for emphasis, new terms, and book titles	Refer to the *LynxOS User's Guide.*

Environment variables, file names, functions, methods, options, parameter names, path names, commands, and computer data
Commands that need to be highlighted within body text, or commands that must be typed as is by the user are **`bolded`**.
	`ls`
	`-l`
	`myprog.c`
	`/dev/null`
	`login:` **`myname`**
	`#` **`cd /usr/home`**

Text that represents a variable, such as a file name or a value that must be entered by the user	`cat` *`filename`*
	`mv` *`file1 file2`*

Blocks of text that appear on the display screen after entering instructions or commands	`Loading file /tftpboot/shell.kdi`
	`into 0x4000`
	`.....................`
	`File loaded. Size is 1314816`
	`Copyright 2002 LynuxWorks, Inc.`
	`All rights reserved.`
	
	`LynxOS (ppc) created Mon Jan 17`
	`17:50:22 GMT 2002`
	`user name:`

Special Notes

The following notations highlight any key points and cautionary notes that may appear in this manual.

NOTE: These callouts note important or useful points in the text.

CAUTION! Used for situations that present minor hazards that may interfere with or threaten equipment/performance.

Technical Support

LynuxWorks Technical Support is available Monday through Friday (holidays excluded) between 8:00 AM and 5:00 PM Pacific Time (U.S. Headquarters) or between 9:00 AM and 6:00 PM Central European Time (Europe).

The LynuxWorks World Wide Web home page provides additional information about our products and LynuxWorks news groups.

LynuxWorks U.S. Headquarters

Internet: `support@lnxw.com`
Phone: (408) 979-3940
Fax: (408) 979-3945

LynuxWorks Europe

Internet: `tech_europe@lnxw.com`
Phone: (+33) 1 30 85 06 00
Fax: (+33) 1 30 85 06 06

World Wide Web

`http://www.lynuxworks.com`

CHAPTER 1

Product Overview

Introduction

These *LynxOS 4.0 Release Notes* provide information on the following topics:

- Product Overview
- New Features

New Features

The following list presents several new features included in LynxOS 4.0:

- New TCP/IP stack based on FreeBSD 4.2
- Linux ABI (Application Binary Interface) compatibility. This feature allows Linux binary applications to run on LynxOS.
- ELF Shared Library Support
- New GNU Toolchain (based on 2.95.3)
- Full `mmap()` POSIX 1003.b support
- Support for AltiVec processor-specific functions and registers.
- Support for customizing core files
- Large RAM support (512 MB) for PPC

For a descriptions of these, as well as additional features, see Chapter 2, “New Features in LynxOS 4.0”.

Supported Board Support Packages (BSP)

The following table describes the supported BSPs for the LynxOS 4.0 release..

Table 1-1: Supported BSPs

BSP Name	Part Number	Boards Supported
`x86_drm`	IA1	x86 boards w/ DRM
`x86_at`	IA1	x86 boards w/o DRM
`cpci_x86`	IAC	x86 cPCI w/ DRM, CPV5350
`cpci_drm`	MC1	Motorola MCP 750, MCPN 750
`mcpn765`	MC3	Motorola MCPN 765 (750, 7400)
`mpmc8260_vads`	MV1	Motorola 8260 ADS
`mpmc860`	MM1	Motorola MPMC860
`mvme5100`	MP4	MVME 5100 (750), 5101 (7400)
`pc_drm`	FP1	FORCE 6750
`pc680`	FP3	FORCE 680 750, 7400
`pmc600_drm`	MP6	Motorola PrPMC600 (8240)
`pmc800_drm`	MP5	Motorola PrPMC 800 (7410)
`pp_drm`	MP2	Motorola MVME 2400
`rpxl823`	MPX	Embedded Planet RPXLite 832e
`sandpoint`	MS1	Motorola Sandpoint
`vmpc`	CT4	Thales VMPC6C

Supported Configurations

For complete installation instructions for LynxOS, refer to the *LynxOS Installation Guide*.

Supported Cross Development Hosts

For the LynxOS 4.0 release, the following cross development hosts are supported:

- Windows XP, Windows 2000
- RedHat Linux 7.2
- Solaris 2.7, 2.8

Supported Linux Reference Platform

For this release, the Supported Linux Reference Platform for the Linux ABI Compatibility layer is:

- Linux Kernel 2.4.x
- Linux `glibc` library 2.2.2

All Linux binaries compiled on the Linux Reference Platform can run on LynxOS 4.0. For information on Linux ABI Compatibility, see the "Linux ABI Compatiblity" chapter in the *LynxOS User's Guide*.

Product Documentation

The LynxOS Documentation CD-ROM contains additional documentation in HTML and PDF formats.

Updates to LynxOS documentation are available from the LynuxWorks website: `http://www.lynuxworks.com`

CHAPTER 2 New Features in LynxOS 4.0

The following sections provide an overview of the new features included in LynxOS 4.0.

New TCP/IP Networking Stack

This release of LynxOS includes an new TCP/IP stack, based on FreeBSD 4.2, enhanced for real-time determinisim and performance. LynxOS TCP/IP supports all of the standard capabilities of the FreeBSD 4.2 stack, including:

- Improved security
- Improved performance
- Improved visibility into the stack (`sysctl`)
- Source-level compatibility for device drivers

Supported RFCs are listed in the *LynxOS Networking Guide*.

Networking Driver Compatibility

For this release of LynxOS, the networking driver interface is unchanged. Networking drivers for previous versions of LynxOS should still work, and only need to be recompiled.

New Networking Utilities

The new TCP/IP stack includes several utilities from the FreeBSD 4.2 stack that are new to LynxOS or replace previous LynxOS networking utilities. Refer to the appropriate man page for usage and configuration information.

Table 2-1: New Networking Utilities

Utility	Description	New/Replacement
`divert`	Diverts packets to a port	New utility
`dummynet`	Bandwidth manager and delay emulator	New utility
`faithd`	Allows IPv6 to IPv4 relays	New utility
`gifconfig`	Configure Gateway interface	New utility
`ipfw` `ip6fw`	Sets firewall policies	New utility
`natd`	Network address translation daemon	New utility
`sysctl`	Command line utility to set/query sysctl variables	New utility
`traceroute` `traceroute6`	Traces networking route to destination	New utility
`tun`	Tunneling utility	New utility
`route` `route6d`	Route table management tool	Replaces previous utility
`arp`	Address resolution display and control	Replaces previous utility
`ifconfig`	Configure network interfaces	Replaces previous utility
`netstat`	Display network status	Replaces previous utility
`ping` `ping6`	ICMP echo requests	Replaces previous utility
`tcpdump`	Traffic monitoring	Replaces previous utility
`tftp`	Includes secure tftp extensions	Replaces previous utility

NOTE: IPv6 utilities are included with the IPv6 package for LynxOS. Support for the IPv6 and IPsec protocols are not included with the standard LynxOS package. These components are available for purchase separately. For information on these products, please contact your LynuxWorks sales representative.

Obsolete Components

With the new TCP/IP stack, STREAMS is no longer supported.

Changes in Networking Stack Default Behavior

The new networking stack provides for greater security and control of its operation. The default behavior of the new stack favors security. Certain LynxOS commands that worked in the previous stack now require explicit commands (via sysctl) to enable them. The following details these commands:

- Source Routing

  The default behavior of the new stack is to disallow source routing. To enable source routing for forwarding IP packets, issue the command:

  ```
  # sysctl -w net.inet.ip.sourceroute=1
  ```

  To allow the stack to accept source routed IP packets, issue the command:

  ```
  # sysctl -w net.inet.ip.accept_sourceroute=1
  ```

- ICMP Broadcast/Multicast Echo

  The default behavior of the new stack is to ignore ICMP echo requests for broadcast and multicast addresses. This prevents a system from being used as a “ping amplifier” in denial-of-service attacks.

  To allow the stack to echo such requests, use the following command:

  ```
  # sysctl -w net.inet.icmp.bmcastecho=1
  ```

- IP Forwarding

  By default, the networking stack doesn't allow the system to function as a router. To allow the stack to forward packets from one interface to another, issue the command:

  ```
  # sysctl -w net.inet.ip.forwarding=1
  ```

Tunnelling Support

Tunnelling support, provided with the new tun utility, is enabled on LynxOS by adding the tun.cfg driver to CONFIG.TBL.

Edit the CONFIG.TBL, and after the line I:hbtcpip.cfg add the following line:

```
I:tun.cfg
```

Rebuild the kernel and reboot the system:

```
# make install
# reboot -aN
```

For additional information on using tun, see the tun(4) man page.

PF_PACKET for Raw Ethernet

LynxOS 4.0 includes support for the PF_PACKET protocol. PF_PACKET is a popular protocol used to send and receive raw ethernet packets to and from a device driver. Previous versions of LynxOS only supported AF_RAWETH, a propietary format that provides raw ethernet functionality.

PF_PACKET is an popular standard, used by Linux to support raw ethernet. Though AF_RAWETH is still supported in this release, it is recommended that customers use the new PF_PACKET interface when developing applications that require raw ethernet. PF_PACKET provides the same functionality as AF_RAWETH.

For information on using PF_PACKET, see the packet(4) and netdevice(4) man pages.

Zebra

The Zebra routing package is included in LynxOS 4.0. Zebra is provided as a tar archive on the “Additional Components” CD-ROM.

Zebra installation, configuration and usage information is documented in the *Zebra User’s Guide*. See the LynxOS 4 Documentation CD-ROM for additional information.

Gigabit Ethernet Support

LynxOS 4.0 includes drivers for the Intel 82533 Gigabit Ethernet Boards.

Intel 82553GC Gigabit ethernet controller is supported only for x86 and DRM-based x86 BSPs (`x86_drm` & `cpci_drm`).

Driver Tunables

The following list provides suggestions for tunging the Intel Gigabit Ethernet driver:

- `debug = 1`

  Enables the debug information for this device only.

- `tx_delay = <`*`value`*`>`

  Should be more than 0, This defers transmitting complete interrupts for `tx_delay` x 1.024 micro seconds.

- `rx_delay = <`*`value`*`>`

  Should be more than 0, This defers receipt frame interrupt for `rx_delay` x 1.024 micro seconds.

- `tx_atonce = <`*`value`*`>`

  Should be more than 0. Device attempts to send `<`*`value`*`>` frames in at once.

ELF Shared Library Support

LynxOS 4.0 adds ELF (Extended Linker Format) and SVR4-style shared libraries and application compatibility to LynxOS 4.0 for x86 and PowerPC host platforms. ELF is a binary format designed to support dynamic objects and shared libraries. The default development library for LynxOS 4.0 is ELF. When an ELF application runs, the system dynamically links required shared libraries to the application.

ELF shared library support provides the ability to manage shared libraries independent of application code, facilitating development. Libraries can be updated independently and older libraries can be maintained for backwards compatibility.

CAUTION! Older binaries compiled on previous versions of LynxOS must be recompiled with the new binary format.

Specifying shared libraries

The `LD_LIBRARY_PATH` environment variable specifies a search path for ELF shared libraries. At times, some ELF shared libraries are dependent on other libraries. If these libraries exist on the system in non-default locations, setting `LD_LIBRARY_PATH` allows the ELF binaries to locate required libraries. Search paths in `LD_LIBRARY_PATH` are separated by colons.

Linux ABI Compatibility

LynxOS supports executing dynamically-linked Linux binary applications on LynxOS systems as if they were native LynxOS applications. There is no need to rebuild Linux applications with LynxOS tools, or even access the source code. Linux application binaries can be installed and executed on a LynxOS machine in the same manner as they are installed and executed on a Linux system. The Linux ABI feature adds a new level of flexibility by allowing users to use both Linux and LynxOS binaries in parallel on a single LynxOS system.

For more information, see the chapter “Linux ABI Compatibility” in the *LynxOS User’s Guide*.

The Linux Reference Distribution for LynxOS 4.0 is:

- Linux Kernel 2.4.x
- Linux `glibc` library 2.2.2

Kernel Changes for Linux ABI Support

Several modifications have been made to the LynxOS kernel to provide the basis for Linux ABI Support. Because of the changes in the kernel, previous COFF applications must be recompiled.

Signal number remapping

Some LynxOS signal numbers are remapped to Linux values.

ioctl() Number Remapping

Existing LynxOS `ioctl()` commands are updated to match those in Linux.

Many Linux `ioctl()` commands use pointers to structures in other arguments containing an unsupported layout structure. These `ioctl()` commands are unsupported.

File Flag Number Remapping

Many of the flags used in the `open()` and `fcntl()` system calls have different values in Linux from those in LynxOS. These system call definitions have changed.

New System Call Mechanism

The new system call mechanism uses software interrupt `0x81`, in the same fashion that Linux uses software interrupt `0x80`. This replaces the older mechanism that used the Long Call 8 instruction. Applications that rely on the old Long Call 8 system call mechanism, or the `0x80` software interrupt do not function.

Any application that makes direct system calls must be rewritten.

New System Calls

For a detailed description of new system calls in LynxOS 4.0, please refer to the corresponding man pages. New system calls in LynxOS 4.0 include:

- `fchdir()`

  `fchdir()` is identical to `chdir()` (change working directory), except that the directory is given as an open file descriptor, similar to `chown()` and `fchown()`

- `setresuid() and setresgid()`

  These system calls set real, effective, and saved user ID and group ID of current processes.

- `wait4()`

  Wait for a child process to exit with full control of all parameters available in the `wait()`, `waitpid()`, and `wait3()` system calls.

mmap() Overview

The virtual memory subsystem of LynxOS 4.0 supports `mmap()` and `munmap()` file memory allocation functions.

`mmap()` can be used to create a mapping between a file or object and process memory address space. This allows processes to access a file or object directly in user memory. Processes do not need to use I/O system calls or access memory in kernel address space.

Refer to the `mmap()` and `munmap()` man pages for detailed usage information.

NOTE: Because of the new `mmap()` functionality, the `smem_create()` function is no longer supported. The functionality of `smem_create()` can be duplicated with `mmap()` and `/dev/mem`. See “Using mmap() to Create smem_create() Functionality” on page 13.

Using mmap

The `mmap()` system call allows you refer to the contents of a file as if it were in memory. For example, if you have a regular file of ascii characters named `myfile.txt`, you can open the file and map it with `mmap()`. The value returned is a virtual address that refers to the first byte of the file. The rest of the file (or any subset of the file that you choose) are mapped linearly to subsequent addresses. Using simple pointer arithmetic you can compute an address that refers to any byte of the file.

Consider the function `display_myfile()`. It opens a file named myfile.txt and uses `lseek()` to find the length of the file. Then, it uses `mmap()` to map the entire file. The address returned by `mmap()` points to the first byte of the file. The for loop outputs each character of the file to standard output.

```
int display_myfile()
{
   long address;
   long file_length;
   int mem_protections;
   int mapping_flags;
   int fd;
   off_t offset_within_file;
   char *c;

   fd = open(“myfile.txt”, O_RDONLY);
   file_length = lseek(fd, 0, 2);

   address = mmap(0, file_length, PROT_READ, MAP_PRIVATE, fd, 0);
   for( p=address; p<address + file_length, p++)
        putchar(*p);
}
```

msync() Coordinates Between Mapped and Read/Write I/O

The POSIX `msync()` system call was not required for previous versions of LynxOS. It now exists, as called for in the POSIX standard. For 4.0, if one process maps a file with `mmap()`, and another process accesses the same file with `write()`, the `mmap()` process must call `msync()` after each write (i.e. each assignment statement changing the file) to ensure coherency of the file system’s buffer cache.

No Device Mapping with mmap()

Some implementations of `mmap()` permit mapping devices as well as mapping files. LynxOS 4.0 does not support device mapping.

Using mmap() to Create smem_create() Functionality

In previous releases of LynxOS, the function smem_create() was used to create a shared mapping from kernel virtual address space to a process virtual address space. This was most often used to allow a process to map resources owned by a device, such as the device registers or private memory. At other times smem_create() was used to allow a process to map kernel memory. The function smem_create() is no longer supported in LynxOS 4.0.

The smem_create() functionality can be duplicated using mmap() in conjunction with the /dev/mem device.

smem_create() took a kernel virtual address as one of its input arguments and returned a process virtual address. The function mmap() can do the same, when it maps from the /dev/mem device.

The following program, mem, is an illustration of the use of /dev/mem and mmap() together. It allows you to display the contents of any readable memory or device, given the virtual address and number of bytes to display.

Note the following:

- The /dev/mem device is opened, returning a file descriptor, fd.
- fd is used as the file descriptor argument in mmap().
- The other arguments to mmap() cause the requested length (len) to be mapped, beginning at the requested kernel virtual address (off), with read-only protection (prot). See the man page for mmap() for a details.

```
#include <sys/file.h>
#include <sys/mman.h>

main(int argc, char *argv[])
{
     int fd, i;
     void *addr = 0;
     unsigned char * pa;
     size_t len = atoi(argv[2]);
     int prot = PROT_READ, flags = MAP_SHARED;
     off_t off = catoi(argv[1]);;

     if (argc < 2) {
          printf("usage: mem <physical address> <len>\n");
          exit (4);
     }
     if ((fd = open("/dev/mem" , O_RDWR)) < 0){
          perror ("open");
          exit (1);
     }
     if ((pa = mmap (addr, len, prot, flags, fd, off) ) < 0) {
          perror("mmap");
          exit(2);

     else {
          printf("Phy address = 0x%x\n", off);
          printf("Length = 0x%x\n", len);
          printf("Mapped virtual address = 0x%x\n", pa);
     }

     for (i = 0; i < len; i++) {
          if (!(i % 16))
               printf("\n");
          printf("0x%x ", *pa++);
     }
     printf("\n");
}
```

GNU 2.95.3 Toolchain

LynxOS 4.0 includes the GNU 2.95.3 toolchain for building user applications. The compilers, linker and assembler creates programs in the ELF object format. Older source code built on previous versions of LynxOS must be recompiled. Software build instructions have not changed, so user’s command arguments and Makefiles should still be valid.

NOTE: The new toolchain is used for creating user applications only. The LynxOS kernel, including device drivers still require COFF (x86) or XCOFF (ppc) formatted executable files compiled with the Cygnus GNU 98r2 tools.

Although the development tools have changed to produce object files in ELF format, they have not changed in functionality. Makefiles and command lines from LynxOS 3.1.0 are still valid for LynxOS 4.0. It is not necessary to change the build procedure to be compatible with LynxOS 4.0. The same command arguments used for controlling the compiler, linker, assembler, etc. work the same as before.

You only need to change the build process if you want to take advantage of new features.

Additions to the GNU 2.95.3 Toolchain

The LynxOS GCC utility includes two options that are used for creating multi-threaded applications, and supporting shared libraries:

`-mthreads`

This option creates an object that supports multithreads.

`-mshared`

This option creates an object that supports shared libraries.

LynxOS 4.0 Kernel Toolchain

The new GNU 2.95.3 toolchain is used for user applications only. The LynxOS kernel, including all device drivers, still requires X/COFF-formatted executable files compiled with the Cygnus GNU 98r2 tools. LynxOS 4.0 includes two complete tool chains, one for application programs and one for the kernel and device drivers.

Setting the Toolchain PATH

Toolchain programs, such as the compiler or linker, are normally invoked with the appropriate command (`gcc` or `ld`, for example). The `PATH` environment variable is set so that the correct program is found. In native development environments, this results in finding the tool in the `/bin` directory. In cross development environments, the environment variable `ENV_PREFIX` refers to the base of the LynxOS tool chain, and the tool program is found relative to that base directory.

It is important to understand that this configuration always finds the tools used for building application programs. Since there are two toolchains, users need to take care not to run the wrong tools.

It is recommended that you use the Makefiles provided by LynuxWorks to build kernels and drivers. This ensures that the correct tools and libraries are used. In some cases, users may want to compile a device driver by hand. To do so, the paths to the appropriate toolchain must be provided. The following tables provide the correct tool paths for each LynxOS installation, for x86 and PowerPC platforms.

Table 2-2: LynxOS 4.0 x86 Kernel Toolchain Paths

Environment	Path to Toolchain
Native	`/usr/i386-coff-lynxos/usr`
Solaris	`$ENV_PREFIX/cdk/sunos-coff-x86/usr/bin`
Windows	`$ENV_PREFIX/cdk/win32-coff-x86/usr/bin`
Linux	`$ENV_PREFIX/cdk/linux-coff-x86/usr/bin`

Table 2-3: LynxOS 4.0 PowerPC Kernel Toolchain Paths

Environment	Path to Toolchain
Native	`/usr/ppc-xcoff-lynxos/usr`
Solaris	`$(ENV_PREFIX)/cdk/sunos-xcoff-ppc/usr/bin`
Windows	`$(ENV_PREFIX)/cdk/win32-xcoff-ppc/usr/bin`
Linux	`$(ENV_PREFIX)/cdk/linux-xcoff-ppc/usr/bin`

Compiling with the Wrong Toolchain

Since applications and kernel code use different libraries, it is not possible to link either kind of object with the wrong tool chain -- any such error would fail due to the unavailability of the required libraries. If you mistakenly use the wrong tool chain, the build will fail immediately.

AltiVec Support

AltiVec-specific functions are supported in the release of LynxOS 4.0. To enable AltiVec functions, users must compile their source code with the switch `-fvec` or `-mvec`.

- For `gcc`, use the `-fvec` switch

  This enables C/C++ support for vector types and vector function as listed in the *Programmer's Interface Manual* (PIM). If invoked with `-fvec`, GCC implicitly passes `-mvec` to the assembler.

- For `as`, use the `-mvec` switch

  This enables AltiVec-specific mnemonics and registers in the assembler as described in the *Programmer's Environment Manual* (PEM).

AltiVec programming documents, including the *AltiVec Programmer's Interface Manual* (PIM) and *Programmer's Environment Manual* (PEM) describe the AltiVec functions and definitions. These documents are accessible from http://www.altivec.org.

AltiVec Registers in GDB

AltiVec processors include a new set of registers called "Vector Registers." GDB allows the display and override of the content of vector registers. Users can print the value as a 128-bit hexadecimal number:

```
(gdb) info registers v0
v0 0x00000011000000120000001300000014
```

or

```
(gdb) print $v0
  $2 = 0x00000011000000120000001300000014
```

Users can also display the vector value as an integer array with four elements, where the format switch of the print command controls how the individual elements are printed:

```
(gdb) print/x $v0
$3 = {0x11, 0x12, 0x13, 0x14}
```

```
(gdb) print/d $v0
$4 = {17, 18, 19, 20}
(gdb) print/c $v0
$5 = {17 '\021', 18 '\022', 19 '\023', 20 '\024'}
```

The content of a vector register can be set similarly as if it were a 16-byte long array.

```
(gdb) print $v0={ 0x15, 0x16, 0x17, 0x18 }
  $7 = 0x00000015000000160000001700000018
```

Or by elements:

```
(gdb) set $v0[2] = 0x12345678
```

Configurable Core File

LynxOS 4.0 provides the ability to configure the information saved in core files. This allows users to control the size of the core file. This feature can be useful in systems with limited memory or disk resources by storing only the data essential for debugging the application at hand.

For more information on creating a configurable core file, see the chapter “Customizing the Default LynxOS Kernel Configuration” in the *LynxOS User’s Guide*.

Configurable Tick Timer

Users can configure the number of ticks per second for real-time clocks. The define `TICKSPERSEC` in the `/usr/include/conf.h` file defaults to 100 ticks per second.

LynuxWorks recommends the following minimum and maximum ticks per second:

Table 2-4: Recommended Ticks Per Second value of TICKSPERSEC

Min ticks per second	Max ticks per second
20 (50ms between ticks)	500 (2ms between ticks)

This number can vary depending on a system's hardware limitations.

Large RAM support

LynxOS 4.0 supports 512 MB of RAM for PowerPC platforms. The following table describes the changes to the PowerPC memory Map.

Table 2-5: Virtual Memory Map for PPC Large RAM Support

Name	Old Value	New Value	Description
PHYSBASE	0xF0000000 256M	0xE0000000 512M	512MB RAM support
IO BASE_1/VME SPACE	0xD0000000 256M	0xD0000000 256M	CPCI/VME space (No change)
IO BASE_2/VME SHORTIO	0xC0000000 256M	0xC0000000 256M	CPCI/VME space (No change)
IO BASE	0xE0000000 256M	0xB0000000 256M	System devices, ISA, PCI-IO, PCI-MEM, Video etc
PERLIMIT	0xF0000000 64MB (grows downwards)	0xB0000000 64MB (grows downward)	
OS BASE	0xB0000000 256MB	0xA0000000 192MB	Top 64MB for PERMAP, OSLIMIT reduced to 192MB
SPECPAGE/Kernel Stack	0xA0000000 256MB	0x90000000 256M	Per-process/Per-thread kernel space

Table 2-5: Virtual Memory Map for PPC Large RAM Support (Continued)

Name	Old Value	New Value	Description
User Memory User text, data, stack, heap, shared	0x10000000 1.75 GB (End: 0x9fffffff)	0x00000000 2.25GB (End: 0x8fffffff)	User space
TRAP PAGES	0x0000 8k	Inside OSBASE (Size 12KB)	Temporary save area

CAUTION! All custom device drivers on PowerPC platforms must be recompiled for this release of LynxOS.

Adaptec Ultra 160 & 2940 U2W SCSI Boards Supported

LynxOS 4.0 includes support for the Adaptec Ultra 160 series of SCSI adapters, including:

- Adaptec Ultra 19160
- Adaptec Ultra 29160
- Adaptec Ultra 29160N
- Adaptec 2940 U2W
- Adaptec 2940 UW

The Adaptec Ultra160 cards are only supported in SE (single-ended) mode and not in LVD (Low-Voltage Differential) mode. Connecting any device to the LVD connector of the SCSI card will not work. However a LVD device can be used with this driver if it is connected to the SE connector.

These boards are supported with the `ascsi` driver. Other Adaptec 2940 SCSI adapters (including the 2940 UW PRO) are supported with the `2940` driver.

Note that the Adaptec Ultra 39160 SCSI Adapter is not supported in this release.