ChorusOS 4.0.1 MPC8xx Target Family Guide

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900
U.S.A.

Part Number 806-5560–10
July 2000

Adobe PostScript

Please Recycle

Contents

CHAPTER **1**

ChorusOS 4.0.1 MPC8xx Target Family Guide

This guide describes how to run the ChorusOS™ 4.0.1 product for the MPC8xx processor family.

Preface

How This Guide is Organized

ChorusOS MPC8xx specific information is provided in the following major sections:

- "Development Environment" on page 8, includes supported hosts, host operating systems and development systems.
- "ChorusOS Supported Features" on page 9, includes kernel components and POSIX components.
- "Libraries" on page 12.
- "Utilities" on page 13, includes host and target utilities.
- "Reference Hardware" on page 16, includes supported reference platforms, supported devices, and validated reference platforms.
- "How to Build and Boot a System Image on the Target" on page 19.
- Appendix A, presents additional man pages for the MONITOR and JVM features (these man pages are not available for on-line search using the `man` command).
- Appendix B, details the list of Solaris packages in the product components, and the associated part numbers.

Related Books

See the *ChorusOS 4.0 Installation Guide for Solaris Hosts* for a description of the installation process of the ChorusOS product on a host workstation running the Solaris™ operating environment. This document also describes how to set up a boot server running the Solaris operating environment.

See the *ChorusOS 4.0 Introduction* for a complete description of the ChorusOS features.

Typographical Conventions

The following table describes the typographic changes used in this book.

TABLE 1–1 Typographical Conventions

Typeface or Symbol	**Meaning**	**Example**
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls –a` to list all files. `machine_name% you have mail.`

`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name%` **`su`**
		`Password:`

AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type **`rm`** *filename*.
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide*. These are called *class* options. You must be *root* to do this.

Shell Prompts

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE 1–2 Shell Prompts

Shell	**Prompt**
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	`#`

Ordering Sun Documents

Fatbrain.com, an Internet professional bookstore, stocks selected product documentation from Sun Microsystems, Inc.

For a list of documents and how to order them, visit the Sun Documentation Center on Fatbrain.com at `http://www1.fatbrain.com/documentation/sun`.

Accessing Sun Documentation Online

The docs.sun.com^{SM} Web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is `http://docs.sun.com`.

Obtaining Technical Support

Sun Support Access offerings are available exclusively to members of the Sun Developer Connection Program. To get free membership in the Sun Developer Connection Program, go to `http://www.sun.com/developers`. For more information or to purchase Sun Support Access offerings, visit: `http://www.sun.com/developers/support` or contact the Sun Developer Connection Program office near you.

Development Environment

The ChorusOS product provides a host-target development environment. Applications are developed on a workstation (the host), and then downloaded and executed on a specific board (the target).

A cross development system is needed to build the applications that execute on the target board (see Section “Utilities” on page 13).

Solaris™ (SPARC™ Platform Edition) Reference Host Environment

Prerequisites for the Solaris host reference configuration are the following:

- Sun SPARCstation™.
- Solaris 2.6, or Solaris 7 (32-bit).
- Sun WorkShop™ 5.0 native compiler.

Note - In order for the CC compiler to work properly, all patches related to the CC compiler must have been installed on the Solaris system.

- JDK™ 1.1.8, for the installation tool.
- JDK 1.2, for the graphical configuration tool and for Java applications.

Cross Compiler

This development environment component is bundled with the ChorusOS for MPC8xx product:

- Chorus Cross Development System 5.0, target PowerPC ELF.

The Chorus Cross Development System is based on the Experimental GNU Compiler System egcs 1.1.2 and binutils 2.9.1 and additional patches.

Graphical Debugger

This development environment component is bundled with the ChorusOS for MPC8xx product:

- XRAY Debugger from Mentor Graphics, ELF format, version 4.4crd.

ChorusOS Supported Features

The following table shows the ChorusOS kernel and operating system optional features that are available for the MPC8xx processor family. The availability status of a feature can be one of:

Y	The feature is supported, and is configurable using the `configurator`(1CC) command, or with the *ews* GUI configuration tool. Please refer to the note at the end of the table for information about specific conditions, or restrictions, for a given supported feature. Some of the features (such as MSDOSFS, FLASH, FS_MAPPER, for example) require specific low-level drivers. These features operate only on platforms which provide these drivers.
N	The feature is not supported.

Feature Description	Feature Name	Availability
Actor management		
Dynamic actor loading management	ACTOR_EXTENDED_MNGT	Y
User-mode extension support	USER_MODE	Y
Dynamic libraries	DYNAMIC_LIB	Y [1]
Compressed file management	GZ_FILE	Y
Scheduling		
POSIX round-robin scheduling class	ROUND_ROBIN	Y
Memory management		

Feature Description	Feature Name	Availability
Virtual (user and supervisor) address space	VIRTUAL_ADDRESS_SPACE	Y
On-demand paging	ON_DEMAND_PAGING	N
Hot restart and persistent memory		
Hot restart	HOT_RESTART	Y
Inter-thread communication		
Semaphores	SEM	Y
Event flag sets	EVENT	Y
Mutual exclusion lock supporting thread priority inversion avoidance	RTMUTEX	Y
Monitors	MONITOR	Y
Time management		
Periodic timers	TIMER	Y
Thread and actor virtual timer	VTIMER	Y
Date and time of day	DATE	Y
Real-time clock	RTC	N
Inter-process communication		
Location-transparent inter-process communication	IPC	Y
Remote (inter-site) IPC support	IPC_REMOTE	Y
Remote IPC communications medium	IPC_REMOTE_COMM	Y
Mailbox-based communications mechanism	MIPC	Y
POSIX 1003.1-compliant message queues	POSIX_MQ	Y
POSIX 1003.1-compliant shared memory objects	POSIX_SHM	Y
LAP		
Local name server for LAP binding	LAPBIND	Y
LAP validity-check option	LAPSAFE	Y
Tools support		

Feature Description	Feature Name	Availability
Message logging	LOG	Y
Profiling and benchmark support	PERF	Y
System Monitoring	MON	Y
System debugging	DEBUG_SYSTEM	Y
C_INIT		
Basic command interpreter on target	LOCAL_CONSOLE	Y
Remote shell	RSH	Y
File system options		
Named pipes	FIFOFS	Y
MS-DOS file system	MSDOSFS	Y
NFS client	NFS_CLIENT	Y
NFS server	NFS_SERVER	Y
UFS file system	UFS	Y
I/O management		
Network packet filter	BPF	Y
Swap support	FS_MAPPER	N
Driver for IDE disk	IDE_DISK	N
`/dev/mem`, `/dev/kmem`, `/dev/null`, `/dev/zero`	DEV_MEM	Y
Support for RAM disk	RAM_DISK	Y
Support for FLASH media [2]	FLASH	Y
Virtual TTY	VTTY	Y
Driver for SCSI disk	SCSI_DISK	N
Support for IPC	IOM_IPC	Y
Support for OSI	IOM_OSI	Y
Networking		
Serial link IP	SLIP	Y
POSIX 1003.1g-compliant sockets	POSIX_SOCKETS	Y
Point-to-point protocols	PPP	Y
Local sockets and pipes	AF_LOCAL	Y
Administration		

Feature Description	Feature Name	Availability
ChorusOS statistics	ADMIN_CHORUSSTAT	Y
`ifconfig` administration command	ADMIN_IFCONFIG	Y
`mount` administration command	ADMIN_MOUNT	Y
`rarp` administration command	ADMIN_RARP	Y
`route` administration command	ADMIN_ROUTE	Y
`shutdown` administration command	ADMIN_SHUTDOWN	Y
`netstat` administration command	ADMIN_NETSTAT	Y
JVM		
Java Virtual Machine	JVM	Y

1. Limitation: the binaries making up the executing image of an actor (main program and dynamic libraries) must hold within a 32MB address range. As a consequence, it is not always possible to load dynamic libraries in flat mode or for supervisor actors.
2. Logical-to-Physical block mapping for flash file system support.

Libraries

The ChorusOS operating system provides the elementary libraries indicated in the following list:

ChorusOS embedded library [1]	`libebd.a`
ChorusOS extended library [1]	`libcx.a`
C++ library	`libC.a`
X11 related client libraries (not thread safe)	`libX11.a`, `libXaw.a`, `libXext.a`, `libXmu.a`, `libXt.a`
Specific BSD APIs (not thread safe)	`libbsd.a`
The SunRPC library	`librpc.a`

The mathematical library	`libm.a`
The “embedded” C library [2]	`stdc.a`
The microkernel “visu” library [3]	`visu.a`

1. The `libebd.a`, `libcx.a`, `libm.a` and `libC.a` libraries have been made thread-safe in order to support multithreaded actors.
2. Included in `libebd.a`.
3. This library is provided for the sake of backwards compatibility only. It is not documented. Its use is strongly discouraged.

Utilities

Target Utilities

The following utilities may be run on the target ChorusOS operating system:

chorusStat(1CC)

cp(1CC)

cs(1CC)

date(1CC)

dd(1CC)

df(1CC)

domainname(1CC)

ftp(1CC)

hostname(1CC)

java(1CC)

ls(1CC)

mkdir(1CC)

mkfifo(1CC)

mv(1CC)

netstat(1CC)

nfsstat(1CC)

pax(1CC)

PROF(1CC)

profctl(1CC)

rdbc(1CC)

rm(1CC)

rmdir(1CC)

startjvm(1CC)

touch(1CC)

uname(1CC)

ypcat(1CC)

ypmatch(1CC)

ypwhich(1CC)

arp(1M)

chat(1M)

chorusNS(1M)

chorusNSinet(1M)

chorusNSsite(1M)

dhclient(1M)

disklabel(1M)

flashdefrag(1M)

format(1M)

fsck(1M)

fsck_dos(1M)

ftpd(1M)

inetNS(1M)

inetNSdns(1M)

inetNShost(1M)

inetNSien116(1M)

inetNSnis(1M)

mkfd(1M)

mkfs(1M)

mount(1M)

mount_msdos(1M)

mount_nfs(1M)

mountd(1M)

newfs(1M)

newfs_dos(1M)

nfsd(1M)

portmap(1M)

route(1M)

shutdown(1M)

slattach(1M)

syncd(1M)

sysctl(1M)

telnetd(1M)

umount(1M)

ypbind(1M)

Host Utilities

The following utilities may be run on the host machine:

chadmin(1CC)

chconsole(1CC)

chlog(1CC)

chls(1CC)

ChorusOSMkMf(1CC)

chserver(1CC)

configurator(1CC)

configure(1CC)

ews(1CC)

mkmerge(1CC)

rdbs(1CC)

profrpg(1CC)

Reference Hardware

ChorusOS targets are described in this section from three different points of view:

Reference Processors and BSPs:

This subsection describes the processors on which the ChorusOS product can run, as well as the details of the BSPs included in the delivery.

Reference Target Platforms:

This section describes all the target platforms which can be used as references in the context of Sun support contracts.

Validated Reference Targets:

This section describes the precise platforms used to run the Sun QA tests; this may be useful, in case of bugs, as a hint or guide to help in identifying issues which are closely hardware related.

Reference Processors and BSPs

The ChorusOS system for MPC8xx supports the following processors:

- Motorola PowerPC MPC860.
- Motorola PowerPC MPC821.
- Motorola PowerPC MPC823.

The ChorusOS system for MPC8xx supports the following reference BSP:

- mpc8xxADS Reference BSP.

mpc8xxADS Reference BSP

Systems

The mpc8xxADS reference BSP supports the following boards:

MPC860 FADS – Motorola SPS.

MPC821 FADS – Motorola SPS.

MPC823 FADS – Motorola SPS.

Devices

The mpc8xxADS reference BSP supports the following MPC860/821 FADS on board devices:

Device Id	**ChorusOS Driver**
cpu (time base and decrementer)	sun:powerpc-(tb,dec)-timer
flash (FLASH memory)	sun:bus-amd29xxx-flash
quicc-8xx (QUICC bus)	sun:powerpc-mpc8xx-(bus, quicc)
quicc-8xx/smc-1 (RS232)	sun:-smc-dbglink (console)
quicc-8xx/smc-2 (RS232)	sun:quicc-smc-uart
quicc-8xx/scc-1 (Ethernet 10BT)	sun:quicc-scc-ether
quicc-8xx/scc-2	not supported
quicc-8xx/scc-3	not supported
quicc-8xx/scc-4	not supported

The mpc8xxADS reference BSP supports the following MPC823 FADS on board devices:

Device Id	ChorusOS Driver
cpu (time base and decrementer)	sun:powerpc-(tb,dec)-timer
flash (FLASH memory)	sun:bus-amd29xxx-flash
quicc-8xx (QUICC bus)	sun:powerpc-mpc8xx-(bus, quicc)
quicc-8xx / smc-1 (RS232)	sun:-smc-dbglink (console)
quicc-8xx / smc-2	not supported
quicc-8xx / scc-1 (USB)	not supported
quicc-8xx / scc-2 (Ethernet 10BT)	sun:quicc-scc-ether

Reference Target Platforms

This section describes all the target platforms which can be used as references in the context of Sun support contracts.

MPC821 / 823 FADS (Motorola / SPS)

Type:	Evaluation / Development Board
Processors:	MPC821 / 823 (50 Mhz)
Main memory:	16-32 MB
Devices:	Asynchronous serial ports (38.4 Kbaud), 10BaseT Ethernet, Timers, Flash memory (AMD 29040 chip)
Hardware monitor:	mpc8bug v1.5 (via ADI Port)

MPC860 FADS (Motorola / SPS)

Type:	Evaluation / Development Board

Processor:	MPC860 PowerQUICC (50 Mhz)
Main memory:	16-32 MB
Devices:	Asynchronous serial ports (38.4 Kbaud), 10BaseT Ethernet, Timers, Flash memory (AMD 29040 chip)
Hardware monitor:	mpc8bug v1.5 (via ADI Port)

Validated Reference Targets

This section describes the precise platforms used to run the Sun QA tests:

- MPC821/823 FADS: MPC8xxFADS Rev. PILOT with MPC821/823FADSDB Rev. PILOT.
- MPC860 FADS: MPC8xxFADS Rev. PILOT with MPC860FADSDB Rev. PILOT.

How to Build and Boot a System Image on the Target

▼ Building a ChorusOS System Image

The following procedure assumes that the ChorusOS product has already been correctly installed on the host workstation. See the *ChorusOS 4.0 Installation Guide for Solaris Hosts*.

1. Create and change to a build directory where you will build system images:

```
$ mkdir build_dir
$ cd build_dir
```

2. **Set an environment variable to use with the** `configure`**(1CC) command as a shortcut to the base directory.**

   For example:

Set the environment variable…	To the family-specific product directory. The default value is…
DIR	/opt/SUNWconn/SEW/4.0.1/chorus-mpc860

3. **Make sure your PATH has been set correctly to include the directory** *install_dir/*`4.0.1/chorus-mpc860/tools/host/bin`**, where the default** *install_dir* **is** `/opt/SUNWconn/SEW`**.**

   Also make sure that your PATH includes `/usr/openwin/bin`, which contains the `imake` utility.

4. **Configure the build directory, using the** `configure`**(1CC) command:**

   If you are building from a binary distribution:

```
$ configure -b $DIR/kernel \
$DIR/os \
$DIR/tools \
-s $DIR/src/nucleus/bsp/drv \
$DIR/src/nucleus/bsp/powerpc \
$DIR/src/nucleus/bsp/powerpc/mpc8xxADS \
$DIR/src/iom
```

Note - The above command configures the build directory to include components installed during a “Default Install”. It does not include optional components, such as the X library or code examples, that you may choose to install separately on Solaris host workstations. For example, in order to include everything in your build environment:

```
$ configure -b $DIR/kernel \
$DIR/os \
$DIR/opt/X11 \
$DIR/tools \
-s $DIR/src/nucleus/bsp/drv \
$DIR/src/nucleus/bsp/powerpc \
$DIR/src/nucleus/bsp/powerpc/mpc8xxADS \
$DIR/src/iom \
$DIR/src/opt/examples
```

If you are building from the source distribution, see the *ChorusOS 4.0 Production Guide*.

As a result of configuration, *build_dir* now contains a `Makefile`, which is used to generate the build environment, and a `Paths` file, which specifies paths to files required by, and created in, the build environment.

5. **Generate the build environment:**

```
$ make
```

6. **Configure the system image:**

```
$ configurator -setenv ETHER_ADDR=xx:xx:xx:xx:xx:xx
```

As you enter the commands above, replace *xx*:*xx*:*xx*:*xx*:*xx*:*xx* with the target system Ethernet address.

7. **Build a system image:**

```
$ make chorus
```

The resulting system image file is located in the build directory, *build_dir* and is called `chorus.RAM`.

Note - You can also make a smaller system image that includes only the operating system kernel:

```
$ make kernonly
```

▼ Placing the System Image on the Boot Server

See the *ChorusOS 4.0 Installation Guide for Solaris Hosts* for instructions on how to configure the boot server.

1. **Copy the system image to the boot server.**

   For example, on a Solaris host workstation:

   ```
   $ rcp chorus.RAM boot_server:/tftpboot
   ```

2. **Verify that everyone has at least read access to the system image on the boot server.**

   For example:

```
$ rlogin boot_server
Password: password_for_user
$ ls -l /tftpboot/chorus.RAM
-rwxr-xr-x   1 user    group     1613824 Dec 15 17:33 chorus.RAM*
```

3. **While logged in to the boot server, create a configuration file for the target.**

   For a target system with IP address `129.157.173.199` using a boot server with IP address `129.157.173.144`, the configuration file contains the following:

```
AUTOBOOT=YES
   BOOTFILE=chorus.RAM
   BOOTSERVER=129.157.173.144
```

   The configuration file is named `/tftpboot/819DADC7.ChorusOS.4.0`, which is constructed from the target system IP address `129.157.173.199` as a concatenation of the following:

   - `129` in decimal translates to `81` in hexadecimal.
   - `157` in decimal translates to `9D` in hexadecimal.

- `173` in decimal translates to `AD` in hexadecimal.
- `199` in decimal translates to `C7` in hexadecimal.
- (optional) `.ChorusOS.4.0` identifies the release, and is appended to the concatenation of the IP address expressed in hexadecimal.

Note - The system first attempts to find the configuration file with the `.ChorusOS.4.0` extension. If it fails to find one, however, it attempts to find a configuration file without the `.ChorusOS.4.0` extension.

How to Boot the Target System Using Motorola's mpc8bug Host Debugger

▼ Booting with the mpc8bug Host Debugger

The following procedure concerns MPC8xx[F]ADS target systems reference platforms.

1. **Install Motorola's ADI host debugger (**mpc8bug**) on the boot host.**

   The *MPC8XX ADS Software (MPC8bug v 1.5)* package contains the `mpc8bug` monitoring tool, configuration files for all reference targets, and documentation explaining how to install and use the tool. This package is provided with the hardware.

   See the mpc8bug installation manual for details about installation.

2. **Append a "ChorusOS load" macro to the .mpctcl.cfg configuration file.**

   The following TCL macro should be appended at the end of the `.mpctcl.cfg` file that is provided as part of the mpc8bug package:

```
a chload reset :h \; load \$1 \; rms der decie 0 \; rms der extie 0 \; rms der prie 0 \;
rms der sysie 0 \; rms der trie 0 \; rms der mcie 0 \; rms der itlbmse 0 \;
rms der dtlbmse 0 \; rms der itlbere 0 \; rms der dtlbere 0 \; go
```

   The above `rms der` *`xx`* `0` commands allow the ChorusOS kernel to handle the coresponding exceptions.

3. **Connect the host ADI and target debug ports.**

   The MPC8xx[F]ADS board must be installed and configured to operate in `Host Controlled` configuration mode via its ADI port. Therefore, you must plug an ADI board into one of the SBus or ISA slots of the boot host and connect to the MPC8xx[F]ADS through a 37-pin flat cable.

See the section “Installation Instructions” in the *MPC8xx[F]ADS User’s Manual* for details about installing the ADI board and configuring the target to boot from the Debug port.

Once this is done, you can start the mpc8bug monitor utility on the boot host as follows:

4. **Connect the RS32 ports of the target system and of the boot system.**

   Connect the target system RS232 port on SMC1 to the boot system serial port for the ChorusOS debug agent link and system console.

5. **Restart the target system.**

6. **Start the** mpc8bug **monitor:**

```
$ mpc8bug ADI ADS
```

Where:

- *ADI* is the number, from `0` to `3`, of the SBus expansion slot for the ADI board on the SPARCstation host system, or the address of the ADI card divided by `0x100` for a PC/AT host.
- *ADS* is the ADI address, from `0` to `3`, of the MPC8xx[F]ADS target board, as determined by the ADDR switches on the DS1 Dip-Switch of the MPC8xx[F]ADS board.

When started, the mpc8bug monitor automatically executes the commands included in the configuration files:

```
mpc8bug version 1.5  May 18 98
Copyright 1998 Motorola, Inc.  All Rights Reserved.

 Initializing memory controller and UPM for 50MHZ
  DRAM delay set to 60ns
  DRAM size set to 16Mbytes
Executing .mpctcl.cfg file from the current directory.
Executing .mpc8xx.cfg file from the current directory.
Executing .mpc860.cfg file from the current directory.
Executing .mpcsdram.cfg file from the current directory.
f860Bug>
```

Make sure the MPC8xx[F]ADS hardware is in a working state by running the diagnostic tests T1, T2, T3 and T4 that are provided by the mpc8bug monitor.

See the mpc8bug documentation for details about using mpc8bug and associated diagnostic programs.

7. **Load the system image through** mpc8bug**:**

```
f860Bug> chload full_path/chorus.RAM
   ( reset :h ; load chorus.RAM ; rms der decie 0 ; rms der extie 0 ; rms der prie 0 ;
rms der sysie 0 ; rms der trie 0 ; rms der mcie 0 ; rms der itlbmse 0 ; rms der dtlbmse 0 ;
rms der itlbere 0 ; rms der dtlbere 0 ; go )
 Initializing memory controller and UPM for 50MHZ
  DRAM delay set to 60ns
  DRAM size set to 16Mbytes
Loading ELF file . . .
Entry point set to 001cf000
Loading section 1 (.ChorusO) : 001ce000 bytes at 00004000
Heap start address set to 00000001
No symbol table
r3 and r5 are set to 0
  Use Ctrl-C to abort execution !
```

The following messages are displayed on the target system console:

```
..... Booting Chorus .....

ChorusOS r4.0.1 for PowerPC - Motorola MPC8xx[F]ADS
Copyright (c) 2000 Sun Microsystems, Inc. All rights reserved.

Kernel modules : CORE SCHED_FIFO SEM MIPC IPC_L MEM_PRM KDB TICK MON ENV ETIMER
LOG LAPSAFE MUTEX EVENT UI DATE PERF TIMEOUT LAPBIND DKI
MEM: memory device 'sys_bank' vaddr 0x7ec22000 size 0x1ce000
/cpu: sun:powerpc-(timebase,dec)-timer driver started
/quicc-8xx: sun:powerpc-mpc8xx-(bus,quicc) driver started
/quicc-8xx/smc-2: sun:quicc-smc-uart driver started
/quicc-8xx/scc-1: sun:quicc-scc-ether driver started
/quicc-8xx/scc-1: Ethernet address 08:00:3e:00:00:06
/quicc-8xx/on-board-flash: error -- wrong (unsupported) CHIP/VENDOR ID
/flash-emul: started as 'One bank of AMD29F040x4 (64k erase block)'
DATE: warning -- svDeviceLookup(rtc) failed (-7)
IOM: SOFTINTR DISABLED (-31). Using an Interrupt thread
IOM Init cluster space from: 0x7ebff000 to: 0x7ec1f800 [65 items of size: 2048]
IOM Init io-buf pool from: 0x7ec1f850 to: 0x7ec1fd70 [8 items of size: 164]
IOM Init raw io-buffer pool from: 0x7ec1fd70 to: 0x7ec211f0 [32 items of size: 164]
Copyright (c) 1992-1998 FreeBSD Inc.
Copyright (c) 1982, 1986, 1989, 1991, 1993
        The Regents of the University of California.  All rights reserved.

max disk buffer space = 0x10000
/rd: sun:ram--disk driver started
/quicc-8xx/on-board-flash: error -- wrong (unsupported) CHIP/VENDOR ID
C_INIT: started
C_INIT: /image/sys_bank mounted on /dev/bd00
C_INIT: found /image/sys_bank/sysadm.ini
C_INIT: executing start-up file /image/sys_bank/sysadm.ini
bpf: ifeth0 attached
IOM: ifnet ifeth0 bound to device /quicc-8xx/scc-1
```

(continued)

(Continuation)

```
bpf: lo0 attached
C_INIT: Internet Address: 129.157.173.199
ifeth0: flags=88437<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet 129.157.173.199 netmask 0xffff0000 broadcast 129.157.255.255
        ether 08:00:3e:00:00:06
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        inet 127.0.0.1 netmask 0xff000000
C_INIT: rshd started
```

How to Boot the Target System from Flash Memory Using bootMonitor

In order to boot the target from flash memory you must perform the following procedures.

▼ Creating a bootMonitor Image

See `bootMonitor`(1CC) for details about how bootMonitor works.

1. **Create a build directory where you will build a** bootMonitor **image:**

```
$ mkdir bootmon
$ cd bootmon
```

   Note that this build directory is different from the directory where you build system images.

2. **Configure the** bootMonitor **build directory based on the binary distribution:**

```
$ configure -b $DIR/kernel \
$DIR/os \
$DIR/tools \
-s $DIR/src/nucleus/bsp/drv \
$DIR/src/nucleus/bsp/powerpc \
$DIR/src/nucleus/bsp/powerpc/mcp8xxADS \
$DIR/src/iom
```

3. **Generate the build environment:**

```
$ make
```

4. **Edit the special** *bootmon*/`conf/mini` **profile so that it reads:**

```
#
# Mini Profile
#

#
# Kernel features
#
-set USER_MODE=false
-set VIRTUAL_ADDRESS_SPACE=false
-set SEM=false
-set EVENT=false
-set MONITOR=false
-set TIMER=false
-set DATE=false
-set RTC=false
-set PERF=false
-set IPC=false
-set MIPC=false
-set LAPBIND=true # Change this from 'false' to 'true'
-set LAPSAFE=true # Change this from 'false' to 'true'
-set MON=false
-set LOG=false
```

5. **Configure the build environment for** bootMonitor**:**

```
$ configurator -p conf/mini
$ configurator -set BOOT_MODE=ROM
```

(continued)

```
$ configurator -setenv ETHER_ADDR=xx:xx:xx:xx:xx:xx
```

As you enter the commands above, replace *xx*:*xx*:*xx*:*xx*:*xx*:*xx* with the target system Ethernet address.

6. **Build a** bootMonitor **image:**

```
$ make bootMonitor
```

The resulting system image file is located in the build directory, *bootmon* and is called `bootMonitor.ROM`.

▼ Flashing the Target System with the bootMonitor Image

1. **Restart the target system.**

2. **Start the** mpc8bug **tool. See Step 6 on page 24 for details.**

3. **Use the** `loadf` **command to flash the bootMonitor:**

```
f860Bug> reset :h ; loadf full_path/bootMonitor.ROM 0x100000
 Initializing memory controller and UPM for 50MHZ
  DRAM delay set to 60ns
  DRAM size set to 4Mbytes
loadf: Loading ELF file . . .
Loading flash mapped sections to ram memory buffer:
Loading section 1 () : 00039000 bytes at 00100000
Programming flash :00039000 bytes  at 02800000-02838fff
Flash programming completed
```

4. **Power off the target board.**

5. **Unplug the ADI cable.**

▼ Booting the Target System

♦ **Restart the target system.**

See Step 7 on page 25 for an example of the messages displayed on the target system console.

APPENDIX **A**

ChorusOS 4.0.1 for MPC8xx Additional Man Pages

The following man pages are not available for on-line use using the `man` command. They will be integrated with the package of man pages in a later major release of the product.

section 1CC: Host and Target Utilities

java(1CC)
startjvm(1CC)

section 2K: Kernel System Calls

monitor(2K), *monitorInit(2K)*, *monitorGet(2K)*, *monitorNotify(2K)*, *monitorNotifyAll(2K)*, *monitorRel(2K)*, *monitorWait(2K)*

section 5FEA: ChorusOS Features and APIs

JVM(5FEA)
MONITOR(5FEA)

NAME

java – Java interpreter

SYNOPSIS

rsh *target* **arun $JVM_ROOT/bin/java** [`–quit`] [`–rehash` [*ENVAR*=*VALUE*]] [`–viewclasses`] [`–viewthreads`] *classname* [*args*]

FEATURES

JVM

DESCRIPTION

`java` is a target utility.

The `java` command executes Java bytecodes created by the Java compiler, javac, on the host system.

The *classname* argument is the name of the class to be executed and must be fully qualified by including the package in the name, for example:

```
example% rsh target arun JVM_ROOT/bin/java java.lang.String
```

Note that any arguments that appear after *classname* on the command line are passed to the **main()** method of the class.

The bytecodes for the class are put in a file called *classname*`.class` by compiling the corresponding source file with javac. All Java bytecode files end with the filename extension `.class`, which the compiler automatically adds when the class is compiled. The *classname* argument must contain a **main()** method defined as follows:

```
class Aclass {
     public static void main(String argv[]){
     . . .
     }
}
```

The `java` command returns control to the command interpreter as soon as it has succeeded in loading the class. It then executes the **main()** method and exits unless **main()** creates one or more threads. In this case, `java` does not exit until the last thread exits. Note that exiting a class *never* causes the Java Virtual Machine to exit in the context of ChorusOS.

When defining classes, specify their locations using the APP_CLASSPATH environment variable, which consists of a colon-separated list of directories that specifies the path.

OPTIONS

The following options are supported:

`–quit`	Kill all running Java threads and terminate the jvmd actor.
`–rehash`	Reload environment variables. If the environment variables to reload are provided as a set of whitespace-separated

	variable-value argument pairs to the `–rehash` option, the Java Virtual Machine will reload only those environment variables that are specified. Otherwise the `–rehash` option forces the Java Virtual Machine to reload all relevant environment variables. (See *ENVIRONMENT VARIABLES*.)
`–viewclasses`	List all Java currently loaded classes in the jvmd actor.
`–viewthreads`	List all Java threads currently running in the jvmd actor.

ENVIRONMENT VARIABLES

The following environment variables are supported:

JVM_ROOT	Base directory where the Java Virtual Machine is installed. Default value: `/opt/jvm` (as seen from the target system).
JVM_CLASSPATH	Search path for non-verified bootstrap classes and resources. Default value: `${`*JVM_ROOT*`}/classes`.
JVM_LIBPATH	Search path for bootstrap native libraries. Default value: `${`*JVM_ROOT*`}/lib`.
JVM_DEBUG	Enables or diables tracing. Values assigned to this environment variable may be: `none` (no tracing), `all` (full tracing), `loading` (provide traces from internal primordial classloader), `verifying` (provide traces from the verifier), `loading, verifying` or `verifying, loading`. Default value: `none`
JVM_GC	Enables or disables garbage collection according to the mark and sweep method. Values assigned to this environment variable may be either `enable` or `disable`. Default value: `enable`

APP_CLASSPATH	Search path for application classes and resources. Default value: None (user-defined).
APP_LIBPATH	Search path for application native libraries. Default value: None (user-defined).

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

SEE ALSO

startjvm(1CC), *JVM*(5FEA).

NOTES

The jvmd actor behaves somewhat differently from Java Virtual Machines designed for other general purpose operating systems. Refer to *startjvm*(1CC) for more information about the jvmd actor.

NAME

startjvm – run the Java Virtual Machine actor

SYNOPSIS

rsh *target* **arun /jvm/bin/startjvm**

FEATURES

JVM

DESCRIPTION

`startjvm` is a target utility.

The `startjvm` command starts the Java Virtual Machine for ChorusOS (the jvmd actor) initializing it according to the environment variables described in the *ENVIRONMENT VARIABLES* section of *java*(1CC). The jvmd actor is started only once, as all java applications executed on the target system run in the context of that actor.

Note that the `startjvm` command is not located under JVM_ROOT.

After the jvmd actor has been started using the `startjvm` command

```
Main JVM created
```

should appear on the ChorusOS console.

The corresponding command to terminate the jvmd actor is:

```
example% rsh target $JVM_ROOT/bin/java -quit
```

The jvmd actor

The jvmd actor runs on the target system.

The jvmd actor provides the Java Virtual Machine for ChorusOS. It may be terminated using the `–quit` option of the `java` command, but does not terminate simply because no Java applications are running.

The Java Virtual Machine component of ChorusOS is implemented as a single supervisor actor. That single actor holds all Java threads associated with all Java applications running on the target. In other words, all Java applications run in the supervisor space and all Java applications and associated threads belong to a single ChorusOS actor.

The following table indicates what is and is not supported.

SUPPORTED	NOT SUPPORTED
All Java™ 2 Platform, Standard Edition, v1.2.2 application programming interfaces *except AWT*, including the following packages: java.beans, java.io, java.lang, java.math, java.net, java.rmi, java.security, java.sql, java.text, java.util, sun.beans, sun.dc, sun.io, sun.jdbc, sun.misc, sun.net, sun.rmi, sun.security, sun.tools, sunw.io, sunw.util	AWT and packages that depend on AWT
The Java™ Native Interface with native code running in supervisor space	The Java Native Interface with native code running in user space only
Loading of dynamic libraries whose symbols are known to the Java Virtual Machine actor	

The following particularities, limitations and restrictions apply:

Single Supervisor Actor

The Java Virtual Machine runs as a single actor.

All Java applications running on the target depend on this single actor, jvmd. All Java applications must be started after the jvmd actor has been launched, using the *java*(1CC) command.

System.exit() Stub

System.exit() takes no action, and simply returns control to the caller.

Java Virtual Machine implementations for other host platforms allow the developer to use **System.exit()** to terminate the Java Virtual Machine currently running. As the Java Virtual Machine actor for the ChorusOS operating system runs multiple Java applications, **System.exit()** is not designed to terminate the Java Virtual Machine itself.

Java Native Interface

Developers writing applications that use the Java Native Interface must use system calls that are available for use by supervisor actors.

The Java Virtual Machine is implemented as a supervisor actor. Some symbols that can be seen in a user space view of the system are not visible in supervisor space.

Messages

All messages from the Java Virtual Machine are directed to the ChorusOS console.

Developers who need to manage messages in some other way must implement their own mechanisms for doing so, for example, by using inter-process communication or log files written to a file system.

Thread Priority

By default, threads running in the Java Virtual Machine share the same priority scale than other system threads.

Before mapping Java thread priorities to the system priorities, developers may first tune the ChorusOS system to set the maximum priority for Java threads using the `jvm.thread.maxPriority` tunable. This value forces threads running in the Java Virtual Machine to be assigned a lower overall priority than other system threads.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

SEE ALSO

java(1CC), *JVM*(5FEA)

NAME

monitor, monitorInit, monitorGet, monitorNotify, monitorNotifyAll, monitorRel, monitorWait – initialize a monitor; acquire a monitor; release a monitor; wait within a monitor for notification; notify a thread waiting within a monitor; notify all threads waiting within a monitor

#include <sync/chMonitor.h>

KnError **monitorInit**(KnMonitor **monitor*);

KnError **monitorGet**(KnMonitor **monitor*);

KnError **monitorNotify**(KnMonitor **monitor*);

KnError **monitorNotifyAll**(KnMonitor **monitor*);

KnError **monitorRel**(KnMonitor **monitor*);

KnError **monitorWait**(KnMonitor **monitor*, KnTimeVal **timeout*);

FEATURES

MONITOR

DESCRIPTION

A monitor is a synchronization object used to protect shared procedures and data against simultaneous access. Once the monitor is acquired by a thread, the thread can suspend its ownership and wait until it is notified or a timeout occurs.

Monitors are `KnMonitor` structures allocated in memory.

monitorInit() initializes the monitor whose address is *monitor*. The monitor is initialized as unlocked.

Statically allocated monitors can be initialized using the K_KNMONITOR_INITIALIZER macro, which initializes the monitor as unlocked. This macro is used as follows:

```
KnMonitor myMonitor = K_KNMONITOR_INITIALIZER
```

monitorGet() is used by a thread to acquire a monitor. If the monitor is unlocked, it becomes locked by the thread and the caller continues its execution normally. If the monitor is already locked by the current thread, execution also continues normally. If the monitor is locked by another thread, the caller is blocked until the monitor is released.

monitorWait() is used by a thread which has acquired a monitor to relinquish its lock on it, to lie dormant until another thread notifies it using **monitorNotify()** or **monitorNotifyAll()**, or until the amount of time specified by timeout has elapsed, and finally to re-acquire its lock on the monitor.

monitorNotify() is used by a thread which has acquired the monitor specified by *monitor* to notify a thread waiting within **monitorWait()** to resume. The calling thread must then call **monitorRel()** so that the waiting thread may actually resume.

monitorNotifyAll() notifies all threads waiting within **monitorWait()** to resume.

monitorRel() is used by a thread which has acquired a monitor to release it. If threads are blocked behind the monitor, one of them is awakened.

A blocking **monitorGet()** is NONABORTABLE (see `threadAbort`(2K)). **monitorWait()** is ABORTABLE, that is, when a **threadAbort()** is addressed to a waiting thread, it behaves as if its time-out had expired.

RESTRICTIONS

A user application and a supervisor application may not share a monitor.

Conversely, two applications running in the same mode (user or supervisor) may share a monitor by mapping it in both address spaces. Such shared monitors must be dynamically allocated monitors. In supervisor mode, the same address may be used by both applications, but care must be taken to keep the monitor's region allocated because the system may crash otherwise.

RETURN VALUES

Upon successful completion, `0` is returned. Otherwise a negative error code is returned.

ERRORS

K_EFAULT	Some of the data provided are outside the address space of the current actor.
K_EINVAL	`waitLimit` is not a valid `KnTimeVal`.
K_EINVAL	The calling thread is not the current owner of the monitor on *monitorRel*, *monitorNotify*, *monitorNotifyAll*, *monitorWait*.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

SEE ALSO

`mutexGet`(2K), `mutexInit`(2K), `mutexRel`(2K), `CORE`(5FEA), *JVM*(5FEA), *MONITOR*(5FEA), `MUTEX`(5FEA)

NAME

JVM – Java Virtual Machine component

FEATURE SUMMARY

The JVM feature provides support for the Java Virtual Machine component. This feature requires the MONITOR feature to be set.

This feature allows the system to provide support for Java applications using the Java Virtual Machine actor, `jvmd` actor.

API

The JVM feature does not itself export an API.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

SEE ALSO

java(1CC), *startjvm*(1CC), *MONITOR*(5FEA).

NAME

MONITOR – monitors

FEATURE SUMMARY

Monitors are a way of synchronizing concurrent threads. A monitor is a set of functions in which only one thread may execute at a time. It is possible for a thread running inside a monitor to suspend its execution so that another thread may enter the monitor. The initial thread waits for the second one to notify it (for example, that a resource is now available) and then to exit the monitor. By extension to object-oriented languages such as the Java™ language, monitor objects are associated with the set of functions. The functions take a monitor object as argument. Only one thread at a time uses a given monitor object. In this context, the term "monitor" often refers to the monitor object itself.

API

The MONITOR feature API is summarized in the following table:

monitorGet()	Obtains the lock on the given monitor.
monitorInit()	Initializes the given monitor.
monitorNotify()	Notifies one thread waiting in **monitorWait()**.
monitorNotifyAll()	Notifies all threads waiting in **monitorWait()**.
monitorRel()	Releases a lock on a given monitor.
monitorWait()	Causes a thread that owns the lock on the given monitor to suspend itself until it receives notification from another thread.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

SEE ALSO

monitorGet(2K), *monitorInit*(2K), *monitorNotify*(2K), *monitorNotifyAll*(2K), *monitorRel*(2K), *monitorWait*(2K), `mutexGet`(2K), `mutexInit`(2K), `mutexRel`(2K), `MUTEX`(5FEA)

APPENDIX **B**

ChorusOS 4.0.1 for MPC8xx Product Packages and Part Numbers

The tables below list the Solaris packages available in this release and indicate the part number for each distinct product component.

Binary Product — for Solaris Host

Part Number	**CLX401-SG70**
Package Name	**Description**
SUNWewbm	Sun Embedded Workshop for MPC8xx BSP source
SUNWewcd	Sun Embedded Workshop PDF Format Common Documentation
SUNWewch	Sun Embedded Workshop HTML Format Common Documentation
SUNWewcp	Sun Embedded Workshop PostScript Format Common Documentation
SUNWewdm	Sun Embedded Workshop for MPC8xx XRAY Debugger
SUNWewgm	Sun Embedded Workshop for MPC8xx GUI Tools
SUNWewim	Sun Embedded Workshop for MPC8xx IOM source
SUNWewjm	Sun Embedded Workshop for MPC8xx JVM

Part Number	**CLX401-SG70**
Package Name	**Description**
SUNWewkm	Sun Embedded Workshop for MPC8xx Kernel
SUNWewm	Sun Embedded Workshop On-Line Manual Pages
SUNWewom	Sun Embedded Workshop for MPC8xx OS
SUNWewpm	Sun Embedded Workshop for MPC8xx Examples
SUNWewsd	Sun Embedded Workshop PDF Format Specific Documentation
SUNWewsh	Sun Embedded Workshop HTML Format Specific Documentation
SUNWewsp	Sun Embedded Workshop PostScript Format Specific Documentation
SUNWewtm	Sun Embedded Workshop for MPC8xx Build Tools
SUNWewum	Sun Embedded Workshop for MPC8xx Debugger and Profiling Support
SUNWewxm	Sun Embedded Workshop for MPC8xx X11 Library
SUNWewzm	Sun Embedded Workshop for MPC8xx egcs Toolchain
SUNWewcab [1]	ChorusOS 4.0.1 Common Documentation Collection
SUNWewsab [1]	ChorusOS 4.0.1 Target Family Documentation Collection
SUNWewmab [1]	ChorusOS 4.0 Reference Manual Collection

1. Answerbook packages cannot be installed using the graphical installer. See the Sun document *Installing and Administering an AnswerBook2 Server* for a complete description of the AnswerBook2 documentation installation process.

Flite Add-on for Solaris Host

Part Number	**FLT401-SG70**
Package Name	**Description**
SUNWewfm	Sun Embedded Workshop for MPC8xx Flite

Source Add-on for Solaris Host

Part Number	**CLX401-SG70-S**
Package Name	**Description**
SUNWewhm	Sun Embedded Workshop for MPC8xx OS source
SUNWewlm	Sun Embedded Workshop for MPC8xx Kernel source

Documentation for Solaris Host

Part Number	**CLX401-SAA0-D1N**
Package Name	**Description**
SUNWewcd	Sun Embedded Workshop PDF Format Common Documentation
SUNWewch	Sun Embedded Workshop HTML Format Common Documentation
SUNWewcp	Sun Embedded Workshop PostScript Format Common Documentation
SUNWewm	Sun Embedded Workshop On-Line Manual Pages
SUNWewsd	Sun Embedded Workshop PDF Format Specific Documentation

Part Number	**CLX401-SAA0-D1N**
Package Name	**Description**
SUNWewsh	Sun Embedded Workshop HTML Format Specific Documentation
SUNWewsp	Sun Embedded Workshop PostScript Format Specific Documentation
SUNWewcab [1]	ChorusOS 4.0.1 Common Documentation Collection
SUNWewsab [1]	ChorusOS 4.0.1 Target Family Documentation Collection
SUNWewmab [1]	ChorusOS 4.0 Reference Manual Collection