Release Notice
CONVEX Fiber Distributed Data Interface V1.1

Document No. 081-017730-004

February 1993

CONVEX Press
Richardson, Texas
United States of America

Release Notice
CONVEX Fiber Distributed Data Interface V1.1

Printed in the United States of America

Table of Contents

Appendices

List of Figures

1

Release Notice

1.1 Overview

This Release Notice describes the V1.1 release of CONVEX Fiber Distributed Data Interface (FDDI). It is designed to supplement the permanent documentation with information that was developed too late for inclusion. Always refer to this release notice before reporting questions or problems; your questions may be answered here. This release notice also lists fixes and workarounds that may save you time if you encounter a known problem.

The FDDI controller is a RISC-based adapter which provides an FDDI connection for computers using the VMEbus. It connects workstations and computer systems to an FDDI network using fiber optics, and it performs much of the processing of the communications protocol and certain FDDI network management functions.

1.2 Prerequisites

The V1.1 release of FDDI has the following prerequisites:

- You must be installing FDDI on a CONVEX C200, C3200, C3400, or C3800 Series machine.
- Installation on a C200, C3200, C3400, or C3800 Series machine requires ConvexOS V10.0.
- You must be running V5.2 or later of SPU OS.
- You should read the *CONVEX FDDI Installation Procedures* before installing FDDI V1.1.

1.3 Notes and Cautions

This section contains general information and words of caution about the product.

- ConvexOS V10.0 requires a system generation of the software drivers when an FDDI controller is installed.
- The maximum distance between FDDI nodes is 2 Km of fiber.
- The maximum total ring length is 100 Km around the ring with 500 dual-attach station attachments.
- The presence of an Optical Bypass Switch (OBS) is detected only at boot time. If you add an OBS after your machine was booted, the switch will not function unless you boot the machine again.
- The network sometimes may not come up after an *ifconfig up* command is issued. This situation may happen if the command is issued during ring transition. (Ring transition is the process in which another node is either connecting to or disconnecting from the ring.) To overcome this problem, wait for a few minutes then enter the following commands:

  % **ifconfig fd0** *hostname* **down arp netmask 0xffffff00**
  % **ifconfig fd0** *hostname* **up arp netmask 0xffffff00**

  where *hostname* is the local FDDI host name.

  If necessary, repeat these command several times until the following message is output on the console:

  ```
  fd%d: Ring status = ring up
  ```

- In normal operation, the FDDI driver will periodically perform an MIB inquiry to the FDDI controller. While a ring transition is in progress, sometimes the MIB inquiry may fail. If this failure occurs, a hardware reset and error recovery is performed automatically. At the same time warning messages similar to the following will be output on the console:

  ```
  fd%d: get mib attribute fddiPORTStatusGrp index 1 failed
  WARNING: resetting fd%d ....
  fd%d: Ring status = ring up
  ```

- If you install a Special Systems VMEbus-based device driver after installing FDDI, you must re-install the FDDI software.

1.4 Associated Documents

The following document is being published with this release:

* *CONVEX FDDI Installation Procedures, V1.1*

The fd(4) man page is new with this release. The arp(4p), ioconfig(4), and ifconfig(8c) man pages have been modified for this release. These man pages are included in Appendix A, "Man Pages."

1.5 File Changes

To accommodate the FDDI driver, changes must be made to the */ioconfig* file on the SPU disk. Figure shows a typical */ioconfig* file, with the FDDI entry in boldface:

Figure 1–1: Example *ioconfig* file

```
viop 4
   vme 0
    ctlr LAN-007 csr 0xFE00 int 5
       unit 0 type ex
    ctlr DKC-204 csr 0x400 int 2
       unit 0 type DKD-206
       unit 1 type DKD-208
    ctlr DKC-204 csr 0x600 int 3
       unit 0 type DKD-206
   vme 1
    ctlr DKC-203 csr 0x800 int 1
       unit 0 type DKD-214
       unit 1 type DKD-214
    ctlr DKC-203 csr 0xa00 int 2
       unit 0 type DKD-214
       unit 1 type DKD-214
    ctlr LAN-208 csr 0x6000 int 6
       unit 0 type fd
```

1.6 Known Restrictions

FDDI does not support "trailer" encapsulation (per RFC1188). This means that trailer packets will never be generated on a transmit operation. If a trailer packet is received, the packet is dropped and counted as a packet error.

A

Man Pages

This appendix contains the modified arp(4p), fd(4), ioconfig(4), and ifconfig(8c) man pages.

NAME

arp – Address Resolution Protocol

SYNOPSIS

pseudo-device ether

DESCRIPTION

ARP is a protocol used to dynamically map between DARPA Internet and network address for 10Mb/s Ethernet as well as 100Mb/s FDDI network. It is used by both of the 10Mb/s Ethernet interface drivers, ex(4) and ve(4), and the 100Mb/s FDDI drivers, fd(4). It is not specific to Internet protocols, to 10Mb/s Ethernet or to 100Mb/s FDDI network, but this implementation currently supports only that combination.

ARP caches Internet-Ethernet address mappings. When an interface requests a mapping for an address not in the cache, ARP queues the message which requires the mapping and broadcasts a message on the associated network requesting the address mapping. If a response is provided, the new mapping is cached and any pending message is then transmitted. ARP will queue at most one packet while waiting for a mapping request to be responded to; only the most recently "transmitted" packet is kept.

To facilitate communications with systems which do not use ARP, *ioctl*s are provided to enter and delete entries in the Internet-to-Ethernet tables. Usage:

```
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <net/if.h>
struct arpreq arpreq;

ioctl(s, SIOCSARP, (caddr_t)&arpreq);
ioctl(s, SIOCGARP, (caddr_t)&arpreq);
ioctl(s, SIOCDARP, (caddr_t)&arpreq);
```

Each ioctl takes the same structure as an argument. SIOCSARP sets an ARP entry, SIOCGARP gets an ARP entry, and SIOCDARP deletes an ARP entry. These ioctls may be applied to any socket descriptor *s*, but only by the super-user. The *arpreq* structure contains:

```
/*
 * ARP ioctl request
 */
struct arpreq {
        struct sockaddr         arp_pa; /* protocol address */
        struct sockaddr         arp_ha; /* hardware address */
        int                     arp_flags;/* flags */
};
/*  arp_flags field values */
#define ATF_COM                 0x02    /* completed entry  */
#define  ATF_PERM 0x04          /* permanent entry */
#define  ATF_PUBL 0x08          /* publish (respond for other host) */
#define  ATF_USETRAILERS 0x10   /* send trailer packets to host */
```

The address family for the *arp_pa* sockaddr must be AF_INET; for the *arp_ha* sockaddr it must be AF_UNSPEC. The only flag bits which may be written are ATF_PERM, ATF_PUBL and ATF_USETRAILERS. ATF_PERM causes the entry to be permanent if the ioctl call succeeds. The peculiar nature of the ARP tables may cause the ioctl to fail if more than 8 (permanent) Internet host addresses hash to the same slot. ATF_PUBL specifies that the ARP code should respond to ARP requests for the indicated host coming from other machines. This allows a host to act as an "ARP server," which may be useful in convincing an ARP-only machine to talk to a non-ARP machine.

ARP is also used to negotiate the use of trailer IP encapsulations; trailers are an alternate encapsulation used to allow efficient packet alignment for large packets despite variable-sized headers. Hosts which wish to receive trailer encapsulations so indicate by sending gratuitous ARP translation replies along with replies

to IP requests; they are also sent in reply to IP translation replies. The negotiation is thus fully symmetrical, in that either or both hosts may request trailers. The ATF_USETRAILERS flag is used to record the receipt of such a reply, and enables the transmission of trailer packets to that host.

Trailer encapsulation is not supported for FDDI network (per RFC1188). Refer to fd(4) for detail.

ARP watches passively for hosts impersonating the local host (i.e. a host which responds to an ARP mapping request for the local host's address).

DIAGNOSTICS

duplicate IP address!! sent from ethernet address: %x:%x:%x:%x:%x:%x. ARP has discovered another host on the local network which responds to mapping requests for its own Internet address.

SEE ALSO

ex(4), ve(4), fd(4), inet(4F), arp(8C), ifconfig(8C)
"An Ethernet Address Resolution Protocol," RFC826, Dave Plummer, Network Information Center, SRI.
"Trailer Encapsulations," RFC893, S.J. Leffler and M.J. Karels, Network Information Center, SRI.
"IP and ARP on FDDI Networks" RFC1188, D. Katz, Network Information Center, SRI.

BUGS

ARP packets on the Ethernet use only 42 bytes of data; however, the smallest legal Ethernet packet is 60 bytes (not including CRC). Some systems may not enforce the minimum packet size, others will.

NOTES

Arp is an optional product; for more information, contact your CONVEX sales representative.

NAME

fd – Interphase 100 Mb/s FDDI network interface

SYNOPSIS

ctlr LAN-208 csr0x%x int %d
unit %d type fd

DESCRIPTION

The *fd* interface provides access to an 100 Mb/s Fiber Distributed Data Interfase (FDDI) Network through an Interphase controller used as a link-layer interface. LAN-208 is the Interphase 4211 FDDI VMEbus controller.

FDDI uses a token ring architecture and employs optical fiber as the transmission medium. It is organized as dual counter-rotating rings to overcome faults in the network through reconfiguration. In normal operation, the primary ring is used for transferring data. When a ring break is detected, the network is mended by channeling data onto the secondary ring.

A special switch called optical bypass switch (OBS) can be used to connect a host to the FDDI ring to provide extra fault tolerance. When the host disconnects itself from the ring or loses power, the OBS bypasses the host and closes the ring automatically.

Each of the host's network addresses is specified at boot time with an SIOCSIFADDR ioctl. The *fd* interface employs the address resolution protocol described in *arp*(4P) to dynamically map between Internet and network addresses (big-endian) on the local network. Big-endian network address and the corresponding hardware network address (used actually in the FDDI packet) are reported on the console right after the *fd* interface is booted. As shown in the following example, each byte in the hardware network address is a bit-swapped counterpart of the big-endian network address.

```
fd%d: address = 00:00:77:01:41:52
fd%d: hardware network address = 00:00:ee:80:82:4a
```

The interface does not support "**trailer**" encapsulation (per RFC1188). This means on transmit, trailer packets will never be generated. If a trailer packet is received, the packet is dropped and counted as a packet error.

When the SIOCSIFFLAGS ioctl is used to mark the interface **down**, ring disconnect will be attempted. If the attempt of ring disconnect fails, a hardware reset to the Interphase controller will be in order. After executing ifconfig **down**, the host is completely removed from the FDDI ring and the following message will be reported

```
fd%d: Ring status = ring down
```

When the SIOCSIFFLAGS ioctl is used to mark the interface **up**, ring connect will be attempted. If the attempt of ring connect fails, a hardware reset and firmware re-downloading to the Interphase controller will be performed. After ifconfig **up** is executed successfully, the host does not actually join to the FDDI ring until the following message is reported

```
fd%d: Ring status = ring up
```

The SMT module of the current Interphase 4211 firmware supports SMT 6.2.2.

SEE ALSO

intro(4n), inet(4F), arp(4P), ifconfig(8)

DIAGNOSTICS

When the remote host is in the process of disconnecting from the FDDI ring or the local host is heavily loaded with network traffic, sometimes the Interphase 4211 controller may stop accepting new transmit request. If this condition happens, a hardware reset and error recovery will be performed automatically. In addition, the following warning message will be reported on the console

fd%d: did not respond to xmit
WARNING: resetting fd%d ...

When the FDDI ring is down momentarily (due to momentarily ring break) and new transmit request is made, the Interphase 4211 controller may abort the transmit request and produce the following message

fd%d: error, xmit response = ring down

When the following error message is output on the console

fd0: error, xmit response = VMEbus error

it implies the Interphase 4211 controller is having difficulty in dealing with transmit request. After this happens, if the controller stops responding to new transmit request, hardware reset and error recovery will be performed automatically.

REFERENCES

ANSI X3T9.5/84-49 FDDI Station Management (SMT) -
Rev 6.2, May 18, 1990
ANSI X3T9.5 FDDI Token RING MEDIA Access Control (MAC) -
Rev 10, Feb. 28, 1986
ANSI X3T9.5 FDDI Physical Layer Protocol (PHY) -
Rev 15, Sept. 1, 1987
ANSI X3T9.5 FDDI Physical Layer Medium Dependent (PMD) -
Rev 7, Feb. 20, 1987
"IP and ARP on FDDI Networks" RFC1188, D. Katz,
Network Information Center, SRI.

NOTES

Fd is an optional product; for more information, contact your CONVEX sales representative.

NAME

ifconfig – configure network interface parameters

SYNOPSIS

/etc/ifconfig *interface* [*address* [*dest_address*]] [*parameters*]
/etc/ifconfig *interface* [*protocol_family*]

DESCRIPTION

ifconfig is used to assign an address to a network interface and/or configure network interface parameters. *ifconfig* must be used at boot time to define the network address of each interface present on a machine; it may also be used at a later time to redefine an interface's address or other operating parameters. The *interface* parameter is a string of the form "name unit", e.g., **ex0**. The address is either a host name present in the host name data base, *hosts*(5), or a DARPA Internet address expressed in the Internet standard "dot notation."

The following parameters may be set with *ifconfig*:

up Mark an interface "up." This may be used to enable an interface after an "ifconfig down." It happens automatically when setting the first address on an interface. If the interface was reset when previously marked down, the hardware will be re-initialized.

down Mark an interface "down." When an interface is marked "down," the system will not attempt to transmit messages through that interface. If possible, the interface will be reset to disable reception as well. This action does not automatically disable routes using the interface.

trailers Request the use of a "trailer" link level encapsulation when sending (default). If a network interface supports *trailers*, the system will, when possible, encapsulate outgoing messages in a manner that minimizes the number of memory-to-memory copy operations performed by the receiver. On networks that support the Address Resolution Protocol (see *arp*(4P); currently, only 10 Mb/s Ethernet), this flag indicates that the system should request that other systems use trailers when sending to this host. Similarly, trailer encapsulations will be sent to other hosts that have made such requests. Currently used by Internet protocols only. Trailer encapsulation is not supported in FDDI (per RFC1188). This flag is ignored by the FDDI driver. Refer to fd(4) for details.

–trailers Disable the use of a "trailer" link level encapsulation.

arp Enable the use of the Address Resolution Protocol in mapping between network level addresses and link level addresses (default). This is currently implemented for mapping between DARPA Internet addresses and Ethernet and FDDI hardware addresses.

–arp Disable the use of the Address Resolution Protocol.

metric *n* Set the routing metric of the interface to *n*, default 0. The routing metric is used by the routing protocol (*routed*(8c)). Higher metrics have the effect of making a route less favorable; metrics are counted as addition hops to the destination network or host.

debug Enable driver dependent debugging code; usually, this turns on extra console error logging.

–debug Disable driver dependent debugging code.

netmask *mask* (Inet only) Specify how much of the address to reserve for subdividing networks into subnetworks. The mask includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask can be specified as a single hexadecimal number with a leading 0x, with a dot-notation Internet address, or with a pseudo-network name listed in the network table, *networks*(5). The mask contains 1's for the bit positions in the 32-bit address that are to be used for the network and subnet parts, and 0's for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion.

dest_address Specify the address of the correspondent on the other end of a point-to-point link.

broadcast (Inet only) Specify the address to use to represent broadcasts to the network. The default broadcast address is an address with a host part of all 1's.

ifconfig displays the current configuration for a network interface when no optional parameters are supplied. If a protocol family is specified, *ifconfig* will report only the details specific to that protocol family.

Only the super-user may modify the configuration of a network interface.

DIAGNOSTICS

Messages indicating the specified interface does not exit, the requested address is unknown, or the user is not privileged and tried to alter an interface's configuration.

SEE ALSO

netstat(1C), intro(4n), rc(8), ex(4), fd(4)

NAME

ioconfig – System I/O configuration description file

SYNOPSIS

The ioconfig file resides in the root of the SPU winchester disk.

DESCRIPTION

The *ioconfig* file contains the description of all the CCU, controller, and peripheral definitions for a given system configuration. It resides in the **SPU** root partition and is used by the operating system and diagnostics for device configuration information. The *ioconfig* file uses the following formats:

For IOP type ccus

iop *ccu_slot_number*
mbus *chassis_number*
ctlr *type* **csr 0x***address* **int** *interrupt_number*
unit *unit no.* **type** *unit_name*

For VIOP type ccus

viop *ccu_slot_number*
vme *chassis_number*
ctlr *type* **csr 0x***address* **int** *interrupt_number*
unit *unit no.* **type** *unit_name*

For 3480-compatible tape, the unit entry is :
unit *unit no.* **subunit** *subunit no.* **type** *unit_name*

For HSP type ccus

hsp *ccu_slot_number*
drvr *type* **csr 0x***address* **int** *interrupt_number*
chnl *channel no.* **type** *unit_name*

For IDC type ccus

idc *ccu_slot_number*
ipi *port_number*
drvr DKC-IP2
unit *unit no.* **type** *unit_name*

ccu_slot_number
A digit which corresponds to the slot number in which a CCU (iop, viop, hsp, or idc) is physically located. In C1 class machines, this will be 3 through 7, inclusive; in C210 class machines, it is 0 through 3.

chassis_number
A number between 0 and 3 for **IOP** ccus, which specifies the multibus chassis connected to the given IOP. For **VIOP** ccus, it is 0 or 1 as the VIOP supports at most two VMEbus chasses. **HSP** type ccus do not have controller chasses and therefore have no chassis_number.

type Type of controller residing in given chassis. For HSP, a user-supplied device controller type. For IDC, a disk device type.

address Control and status register (csr) address in hexadecimal format, corresponding to the board strapping for the specific controller type.

interrupt_number
interrupt number associated with the controller.

unit_name
Device unit specification; the disk type name as specified in the /etc/disktab file.

port_number
The IDC port number, in the range 0 to 3.

unit_no The device unit number. Its range is device-specific. For 3480-compatible tape, the range is 0 to 6 which stands for the SCSI ID of the formatter.

subunit_no
The device subunit number. Its range is device-specific. For 3480-compatible tape , the range is 0 to 3 which stands for the LOGICAL UNIT NUMBER of the 3480-compatible tape drive.

Below is an example of an *ioconfig* file which supports the following: one disk, one LAN interface, and a 16-line terminal controller on Multibus 0 connected to IOP 3; two disks, one LAN interface and one 3480-compatible formatter attached with two 3480-compatible tape drives on VMEbus 0 connected to VIOP 5; two IPI disks on two different ports in a IDC in CCU 6.

```
iop 3
 mbus 0
  ctlr DKC-001 csr 0xdc0 int 2
   unit 0 type DKD-001
  ctlr LAN-001 csr 0xc40 int 1
   unit 0 type ex
  ctlr ACM-001 csr 0x020 int 7
   unit 0 type TTY
   unit 1 type TTY
   unit 2 type TTY
   unit 3 type TTY
   unit 4 type TTY
   unit 5 type TTY
   unit 6 type TTY
   unit 7 type TTY
   unit 8 type TTY
   unit 9 type TTY
   unit 10 type TTY
   unit 11 type TTY
   unit 12 type TTY
   unit 13 type TTY
   unit 14 type TTY
   unit 15 type TTY
viop 5
 vme 0
  ctlr DKC-203 csr 0xc00 int 1
   unit 0 type DKD-214
   unit 1 type DKD-214
  ctlr LAN-007 csr 0x001 int 2
   unit 0 type ve
  ctlr MTC-202 csr 0xee00 int 3
   unit 0 subunit 0 type MTD-207
   unit 0 subunit 1 type MTD-207
idc 6
 ipi 0
  drvr DKC-IP2
   unit 0 type DKD-502
 ipi 1
  drvr DKC-IP2
   unit 0 type DKD-502
```

SEE ALSO

intro(4), info(4)

B

Configuring FDDI

The attached information will be included in the next edition of *Managing Internet Services and NFS*, to be published in late 1993. It is included here for easy reference.

CONVEX Fiber Distributed Data Interface (FDDI)

Before you can use the FDDI, you must complete the following tasks:

1. Install FDDI software.
2. Install and configure FDDI hardware.
3. Add an entry for the FDDI hardware to the /ioconfig file.
4. Customize networking boot-time parameters.
5. Test the hardware configuration.
6. Configure the FDDI using the `ifconfig` command.

This document describes how to perform these tasks.

Installing FDDI software

To install FDDI software, follow the steps outlined in the installation procedures that came with your CONVEX FDDI and CONVEX Internet Services software.

Installing and configuring FDDI hardware

Install and configure FDDI hardware according to the instructions in Chapter 2, "Configuration and Installation," of the *CONVEX Fiber Distributed Data Interface Service Guide*.

Adding FDDI hardware to the /ioconfig file

The /ioconfig file contains a description of all the Channel Control Units (CCUs), interfaces, controller boards, and peripheral devices for your system. The boot process reads this file to determine what devices are present.

This section describes the format of the /ioconfig file, and gives instructions for adding an entry for FDDI. *Managing ConvexOS: Configuration Guide* describes device configuration in detail. You can find supplemental information in Chapter 3, "Integration and Test," of the *CONVEX Fiber Distributed Data Interface Service Guide*, and in the man pages for fd(4) and ioconfig(4).

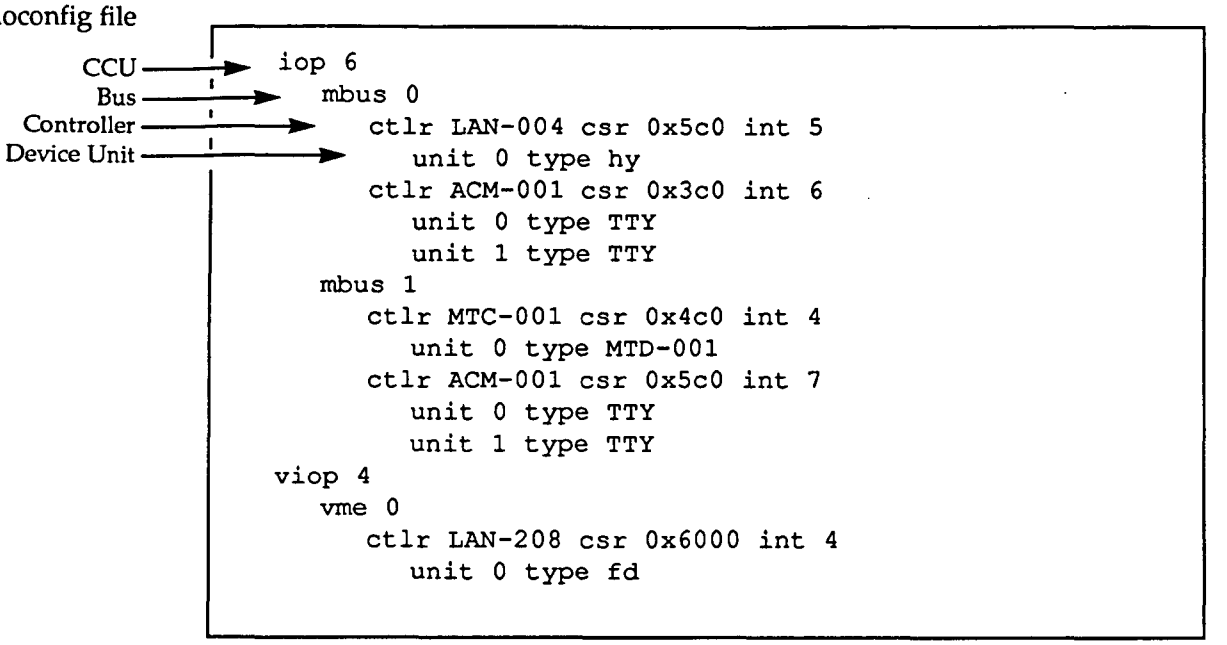
The /ioconfig file

The /ioconfig file is located on the Service Processor Unit (SPU) disk. To display this file, enter the following command:

```
spu -r /ioconfig
```

Figure 1 shows a sample /ioconfig file.

Figure 1
Example /ioconfig file



The /ioconfig file contains an entry for each CCU connected to the system. Each entry contains several levels of information, usually distinguished by indention levels for readability. The following sections describe the different levels of information in the /ioconfig file.

CCU description

Beginning at the left margin, the CCU description identifies the CCU type and slot number. Supported CCU types for network devices include:

- IOP—For multibus Ethernet and HYPERchannel interfaces.
- VIOP—For VME Ethernet, FDDI, HYPERchannel, and UltraNet interfaces.

Slot numbers range from 0 to 7, corresponding to the I/O slot to which the CCU is connected.

Bus or interface description

Information at the first level of indention identifies the type and chassis number of the bus or interface. Supported types for network interfaces are:

- Multibus (mbus) for IOP-type CCUs.
- VMEbus (vme) for VIOP-type CCUs.

IOP- and VIOP-type CCUs each support two chassis, numbered zero and one.

Controller or driver description

At the second level of indention, the controller description specifies I/O controller type, unique control and status register (CSR) address, and interrupt level.

The format for a network interface description is:

`ctlr` *type* **`csr`** *address* **`int`** *level*

where

`ctlr` *type* — Type of controller. Table 1 lists controller types and descriptions.

`csr` *address* — Control and status register (CSR) address in hexadecimal format, corresponding to the board strapping for the specific controller type.

`int` *level* — Controller's interrupt number. It can be any number between 0 and 7 that is not already used.

The CSR address and interrupt level for a controller are provided in the specific controller documentation and in the *CONVEX Guide to Attaching Multibus Peripherals* or the *CONVEX VMEbus Service Documentation Kit.*

Table 1
Network controller types

Controller type	Description
LAN-001	Excelan multibus Ethernet controller
LAN-002	HYPERchannel multibus device driver
LAN-004	HYPERchannel multibus controller
LAN-007	Excelan VMEbus Ethernet controller
LAN-202	UltraNet VMEbus controller
LAN-204	HYPERchannel VMEbus controller
LAN-208	FDDI VMEbus controller

Device unit description

At the third level of indention, the device unit description specifies device unit number and type.

For IOP- and VIOP-type CCUs, the format for the unit description is:

`unit` *number* **`type`** *name*

where

`unit` *number* — is the number assigned to the unit. Each FDDI VMEbus controller (LAN-208) supports a single FDDI device; therefore, each FDDI must be assigned a unit number of 0.

`type` *name* — is the type of unit connected to the controller. Table 2 on the following page lists unit types and descriptions.

Table 2
Network device unit types

Unit type	Description
ex	Multibus or VMEbus Excelan Ethernet interface
HYP-001	Multibus HYPERchannel device driver
hy	Multibus or VMEbus HYPERchannel network interface
unet	VMEbus UltraNet network interface
fd	VMEbus FDDI network interface

Adding an entry for FDDI to the /ioconfig file

After installing FDDI hardware, you must reconfigure ConvexOS to recognize the new device. Edit the /ioconfig file on the SPU to insert the device definition under the appropriate IOP and bus numbers. To edit the /ioconfig file for your system, first boot to the SPU prompt, `(spu)>`. Refer to the *CONVEX Processor Operation Guide* for booting instructions.

For an FDDI, insert an entry similar to that in Figure 2.

Figure 2
VMEbus FDDI /ioconfig entry

```
viop 1
   vme 1
      ctlr LAN-208 csr 0x6000 int 4
         unit 0 type fd
```

Reboot the CONVEX system and bring it up in multiuser mode. This forces the system to autoconfigure the newly-installed network hardware.

Customizing boot-time parameters

This section describes boot-time parameters related to network performance. For a discussion about how to change boot-time parameters, see *Managing ConvexOS: Configuration Guide.*

When you boot your system, the boot process reads a file containing parameters that control the way the operating system handles CPUs and CCUs at your site. You can customize these parameters for the network interface you have installed. Table 3 describes each networking boot-time parameter.

Table 3 Networking boot-time parameters

Parameter	Description
`gateway`	Enables sending ICMP errors if both of the following conditions are true: ❑ The system has a single network interface, or `ipforwarding` is disabled. ❑ The received IP packet is not for the system that has `gateway` enabled. If `gateway` is disabled, under these circumstances, errors are not set to the source machine and the packet is dropped. See `ipforwarding`, `ipsendredirects`, and `subnetsarelocal`. default = 0, off = 0, on = 1
`ipforwarding`	Enables forwarding of IP packets when the IP address does not match any of the Internet addresses for the machine's network interfaces. If this parameter is disabled, packets can be dropped. See `ipsendredirects`, `gateway`, and `subnetsarelocal`. default = 1, off = 0, on = 1
`ipsendredirects`	Enables sending ICMP redirects to inform the sending machine of the direct address to which packets were forwarded. If this option is disabled, redirects are not sent when packets are forwarded. See `ipforwarding`, `gateway`, and `subnetsarelocal`. default = 1, off = 0, on = 1
`subnetsarelocal`	Considers an Internet address local even if it belongs to another subnet. A different network number means the address is remote, that is, accessible through a gateway. This option is used only during determination of the TCP maximum segment size. TCP uses a small maximum segment size (536 bytes) for remote destinations. For local destinations, TCP uses the maximum transmission unit of outgoing network interface. See `ipforwarding`, `ipsendredirects`, and `gateway`. default = 1, off = 0, on = 1
`udpcksum`	Enables checksumming of UDP datagrams. While checksumming adds substantial overhead, turning the parameter off increases the risk of allowing bad packets farther up in the protocols. default = 0, off = 0, on = 1
`fd_max_recv`	Defines the maximum number of outstanding receive requests. default = 4, min. = 2, max. = 18
`fd_max_xmit`	Defines the maximum number of outstanding transmission requests. default = 8, min. = 4, max. = 36

Testing the hardware configuration

Test FDDI hardware according to the instructions in Chapter 3, "Integration and Test," of the *CONVEX Fiber Distributed Data Interface Service Guide.*

Configuring the FDDI

Before running network software, you must configure the FDDI by using the `ifconfig` command.

You must be logged in as `root` to use `ifconfig`. Interfaces are numbered in the order in which they are listed in the /ioconfig file. For instance, the first FDDI is assigned the unit name, `fd0`, the second, `fd1`, and so forth.

The command syntax is:

/etc/ifconfig *interface host_name* **{up | down}**
[netmask *mask***] [broadcast** *address***]**

where:

interface	Name and number of the network interface. (Refer to the fd(4) man page.)
host_name	Local host name; for example, `acme1`, or the internet address in dot notation.
up	Start the network interface. The complementary argument, `down`, shuts down the interface.
netmask	*mask* specifies the portion of the internet address to reserve for the combined network and subnet fields. (Refer to "Defining Subnet Masks," below.)
broadcast	*address* used to represent network broadcasts. By default, the broadcast address has a host part of all ones. (Refer to "Defining Broadcast Addresses," below.)

Note

The `ifconfig` command has more parameters than those discussed here. Refer to the ifconfig(8C) man page for more information.

Run the `ifconfig` command from the command line to ensure that it works properly before you edit /etc/rc.local to run it automatically upon system startup. Verify that the system accepted the information by entering:

ifconfig *interface*

For the specified *interface* (for example, `fd0`), the system displays the host address, network status (up or down), network mask, and so forth.

Defining broadcast addresses

The default broadcast address contains a host part of all ones. You can change the broadcast address for an interface by using the `ifconfig` command. The system accepts broadcasts with a host part of all zeros (for compatibility with systems that use BSD 4.2 broadcasts) but transmits broadcasts with the destination address set to the broadcast address assigned with `ifconfig`.

If a machine on a network does not understand the broadcast address you select, some utilities, such as `rwho`, fail. If this happens, use `ifconfig` to set the network interface broadcast address to the old broadcast address (all zeros) for all machines on the network.

Note

All machines that communicate with each other must use the same broadcast address, either all zeros or all ones. The preferred method is a broadcast address of all ones, as in `broadcast 128.194.255.255`.

Defining Subnet Masks

Ones in the subnet mask indicate bit positions to use for the combined network and subnet fields; zeros mark the positions of bits in the host field. You specify the mask as a single hexadecimal number with a leading 0x, or in dot notation. For example, specifying

```
netmask 0xffffff00
```

or

```
netmask 255.255.255.0
```

both indicate you want 24 bits of combined network and subnet fields, and 8 bits of host number. For a class B network, this mask logically partitions your 16 bits of host number into an 8-bit subnet field and an 8-bit host field. If you do not supply `netmask`, the mask is set according to the network class (A, B, or C with 8, 16, or 24 bits of network part, respectively).

Note

To avoid confusion with broadcast addresses, do not use subnet numbers of all zeros or all ones.

Configuring at start-up

When you are confident that the network is properly configured, add `ifconfig` commands to your /etc/rc.local file for each network interface on your system, so that the network will be brought up at boot time. An example is shown in italics in Figure 3. This line causes the FDDI network to be configured automatically each time the system is started.

Figure 3
Sample /etc/rc.local file

```
#
/bin/hostname acme
#
if [ -f /etc/ifconfig ]; then
        /etc/ifconfig ex0 `/bin/hostname` up trailers netmask 0xffffff00
        /etc/ifconfig fd0 `/bin/hostname` -f up netmask 0xffffff00
        #
        # All physical interfaces MUST be configured before lo0 is configured
        #
        /etc/ifconfig lo0 localhost up
fi
#
# syslogd must be started before any other daemons.
#
if [ -f /usr/etc/syslogd ]; then
        rm -f /dev/log
        /usr/etc/syslogd & echo 'starting system logger'
        if [ -f /usr/adm/bin/errlogd -a -f /usr/local/bin/perl ] ; then
                /usr/adm/bin/errlogd 2>&1 & echo ' errlogd' 2>&1
        fi
fi
#
if [ -f /etc/routed ]; then
        /etc/routed & echo -n ' routed'
fi
#
echo -n 'checking quotas: '
        /usr/etc/quotacheck -p -a
        /usr/etc/quotaon -a
echo 'done.'
#
echo -n 'local daemons:'
#
if [ -f /etc/rwhod ]; then
        /etc/rwhod & echo -n ' rwhod'
fi
#
```

Verifying network configuration with netstat

Bringing the machine up in multiuser mode is good evidence that you have correctly configured network interfaces. Usually, the machine simply does not run in multiuser mode if you make a mistake during the configuration process. Of course, you should test the network after the system is up and running in multiuser mode.

You must configure the network interface as described in this chapter before configuring network utilities according to the procedures discussed in Chapter 8, "Configuring Network Services," of the *CONVEX Internet Services System Manager's Guide.* After you have configured network utilities, you should be able to use the network and its associated utilities. As a check, exercise the utilities associated with each of the daemons you have installed. You can check the network configuration by entering:

```
netstat -i
```

If the network is properly configured, the system displays output similar to that shown in Figure 4.

Figure 4
Configuration check: sample `netstat -i` output

```
Name  Mtu   Network      Address    Ipkts   Ierrs Opkts    Oerrs  Collis   Queue
fd0   4352  acme-net     acme       1520512   0   1327049  0      293      0
lo0   1536  loopback-n   localhost  434254    0   434254   0      0        0
```

The displayed host name and network name are specific to your installation.