C²I² Systems

CCII Systems (Pty) Ltd Registration No. 1990/005058/07

Communications
Computer Intelligence
Integration

User Manual

for the

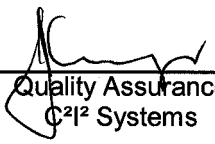
Fibre Channel Adapter

VxWorks Enhanced Network Software Driver

C²I² Systems Document No.	CCII/FC/6-MAN/001
Document Issue	1.2
Issue Date	2009-08-20
Print Date	2009-08-20
File Name	P:\Fibrechn\TECH\MAN\CFCMAN01.WPD
Distribution List No.	DN 0645

Document prepared by C²I² Systems, Cape Town

Signature Sheet

Name	Signature	Date
Completed by pp. X. Kruger	 Project Engineer Board Level Products C²I² Systems	2009-08-20
Accepted by L. DE LANGE	 Project Manager Board Level Products C²I² Systems	2009-08-28
Accepted by X. Kruger	 Quality Assurance C²I² Systems	2009-08-20

Signature Sheet

Name	Signature	Date
Completed by		
	Project Engineer Board Level Products C²I² Systems	
Accepted by		
	Project Manager Board Level Products C²I² Systems	
Accepted by		
	Quality Assurance C²I² Systems	

Amendment History

Issue	Description	Date	ECP No.
0.1	Initial Release.	2002-04-23	-
1.0	Baseline Document.	2002-10-24	-
1.1	Fixed up document and added description of initialisation string.	2005-07-21	CCII/FC/6-ECP/011
1.2	Improve document naming consistency.	2009-08-20	CCII/FC/6-ECP/016

Contents

Abbreviations and Acronyms

BSD	Berkley Software Distribution
BSP	Board Support Package
END	Enhanced Network Driver
FC	Fibre Channel
FDDI	Fibre Distributed Data Interface
FTP	File Transfer Protocol
HCC	Host Carrier Card
ICMP	Internet Control Message Protocol
IP	Internet Protocol
LAN	Local Area Network
MIB	Management Information Base
NIC	Network Interface Card
PC	Personal Computer
PCI	Peripheral Component Interconnect
PMC	Peripheral Component Interconnect Mezzanine Card
SBC	Single Board Computer
SENS	Scalable Enhanced Network Stack
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VME	Versa Module Eurocard

1. Scope

1.1 Identification

This document is the User Manual for the Fibre Channel (FC) Enhanced Network Driver (END) VxWorks Software Driver.

1.2 Introduction

The FC END VxWorks Software Driver was developed to support the Intel X86 PC, MVME5100 PPC and the SVME179 PPC platforms. As such the drivers binaries are provided with explicit installation instructions. The FC END VxWorks Software Driver conforms to VxWorks 5.4 END driver model and the driver will interface to VxWorks using this standard.

The FC Adapter is a dual channel PCI Mezzanine Card to FC protocol controller, attaching computers to 1 Gbit/s FC networks using copper cable and 2 Gbit/s using fibre optic cable.

At present, C²I² Systems FC Adapters are available in PMC formfactor.

The driver software distribution consists of the following files :

ccfcEndxxx.a[1]	FC END VxWorks Software Driver object file
ccFddiShim.o	Shim module for FDDI compatibility
ccFddiShim.c	Shim source file
ccMib.h	Header file for FDDI END VxWorks Software Driver SMT 7.3 MIB
Readme.txt	Installation notes
Release.txt	Release notes and revision history. Please check this file for information on the latest updates.

[1] *xxx*=X86 for Intel X86 PC, DY4PPC for DY4179/181/182 or 5100 for MV5100 PPC

2. Applicable and Reference Documents

2.1 Applicable Documents

2.1.1 RFC2625, *IP and ARP over Fibre Channel* - http://www.ietf.org/rfc/rfc2625.txt, dated June 1999.

2.1.2 DI-IPSC-81443, *Data Item Description for a Software User Manual*, dated Apr 89.

2.1.3 VxWorks 5.5, *Programmer's Guide*, dated 2002-07-23.

2.1.4 VxWorks 5.5, *Network Programmer's Guide*, dated 2002-08-14.

2.2 Reference Documents

None.

3. Getting Started

This section is a quick start guide for getting the FC END VxWorks Software Driver up and running.

3.1 Loading the Driver

1. Boot VxWorks and load the ccfcEndxxx.a[1] lib with '`ld < ccfcEndxxx.a`'.

2. Type '`muxDevStart ( muxDevLoad (unitno, ccfcLoad,"",0,0 ))`' to attach and start the FC END VxWorks Software Driver . Verify by using `muxShow`. The *unitno* must be 0 for Function[0] and 1 for Funtion[1] on the multi-function FC Adapter.

3. Attach the FC END VxWorks Software Driver to the TCP/IP protocol stack. Type '`ipAttach unitno, "ccfc"`' and use `muxShow` and `ifShow` to confirm that the *ccfcunitno* exists and was attached to TCP/IP.

4. Set the IP address using '`ifAddrSet`'. Refer to the VxWorks Network Programmers Guide paragraph '*4.4 Overview of TCP/IP'* for setting up IP host names and IP routing.

5. Confirm the FC LAN connection using ping. An example is shown below.

Note : This example assumes there is already a Network Interface Card (NIC) on the FC LAN which has been set up with an IP address of 10.0.0.1.

```
-> muxDevStart ( muxDevLoad ( 0,ccfcLoad, "", 0, 0 ))
-> ipAttach 0 , "ccfc"

-> ifShow
ppp (unit number 0):
      Flags: (0x71) UP POINT-TO-POINT ARP RUNNING
      Internet address: 172.16.0.2
      Destination Internet address: 172.16.0.1
      Netmask 0xffff0000 Subnetmask 0xffff0000
      Metric is 0
      Maximum Transfer Unit size is 1 500
      5 packets received; 5 packets sent
      0 input errors; 0 output errors
      0 collisions
lo (unit number 0):
      Flags: (0x69) UP LOOPBACK ARP RUNNING
      Internet address: 127.0.0.1
      Netmask 0xff000000 Subnetmask 0xff000000
      Metric is 0
      Maximum Transfer Unit size is 4 096
      0 packets received; 0 packets sent
      0 input errors; 0 output errors
      0 collisions
ccfc (unit number 0):
      Flags: (0x63) UP BROADCAST ARP RUNNING
      Netmask 0xffffffff Subnetmask 0xffffffff
      Ethernet address is 00:50:C2:18:D0:00
      Metric is 0
      Maximum Transfer Unit size is 16 384
      18 packets received; 0 packets sent
      0 input errors; 0 output errors
      0 collisions
-> ifAddrSet "ccfc0","10.0.0.4"
```

[1] *xxx*=X86 for Intel X86 PC, DY4PPC for DY4179/181/182 or 5100 for MV5100 PPC

```
-> ping "10.0.0.1"
PING 10.0.0.1: 56 Data Bytes
64 Bytes from 10.0.0.1: icmp_seq=1. time=0. ms
64 Bytes from 10.0.0.1: icmp_seq=2. time=0. ms
64 Bytes from 10.0.0.1: icmp_seq=3. time=0. ms
```

3.2 Loading the FDDI Shim Module

In some instances, the FC Adapter will be used to replace the FDDI Adapter in an existing system. In this case, there is no need to change the user applications. The only difference between the implementation of the FC and FDDI END VxWorks Software Drivers are that the fddiGetStats() routine is replaced by the ccfcGetStats() routine. The shim is present to restore the difference. Figure 1 below shows a graphical layout of the difference between the two drivers. Users of the FDDI END VxWorks Software Driver will be familiar with the structure **cc_fddi_mib_type** and the following routines :

- **void ccfddiGetStats(struct cc_fddi_mib_type *data)**
- **void ccfddiClrStats(void)**

Although the FC END VxWorks Software Driver is not compatible with the FDDI MIB, calls to the MIB routines will not result in an error as long as the FDDI shim module is loaded. In the FDDI structure cc_fddi_mib_type, some fields may or may not correspond with fields in structures of the FC END VxWorks Software Driver . As per default, all the fields will be set to NULL. If any fields in the FDDI MIB needs to be changed to ensure compatibility with a application, the values can be changed in the `ccFddiShim.c` source file and recompiled. A comparative layout of the difference between the drivers are shown below.

Load the FDDI shim with '`ld < ccFddiShim.o`'.

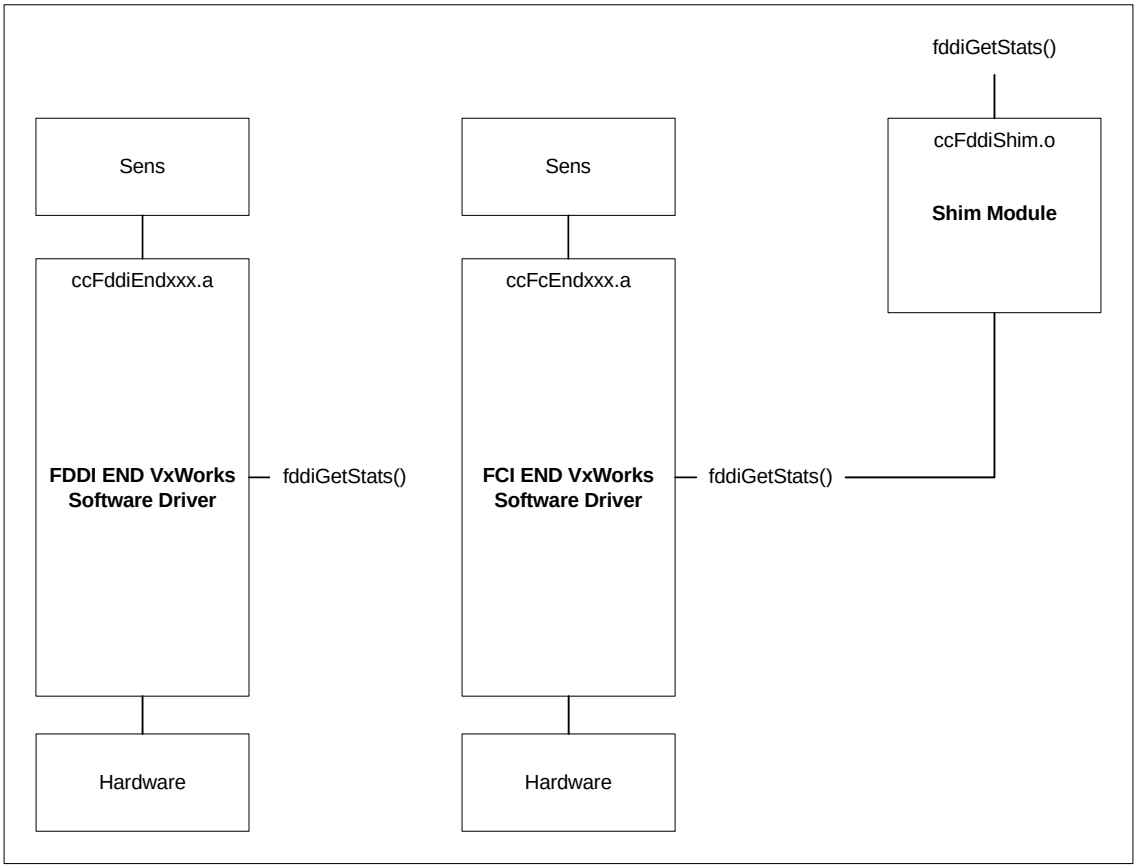


Figure 1 : Comparative Layout of the FDDI and FC END VxWorks Software Drivers

4. Installation Procedure

4.1 Building the FC END VxWorks Software Driver into the VxWorks Kernel

This section describes the installation instructions for building the FC END VxWorks Software Driver into the VxWorks kernel.

1. Copy ccfcEndxxx.a[1] to your BSP library directory (eg /tornado/target/config/X86/lib).

2. Edit the Makefile in the BSP directory (eg /tornado/target/config/X86).

   Find the line
   ```
   MACH_EXTRA  =
   ```

   and replace with
   ```
   MACH_EXTRA  =  ./lib/ccfcEndxxx.a[1] ./lib/ccfcShim.o[2]
   ```

3. Add the following code fragment to config.h.

   (before "#define DEFAULT_BOOT_LINE") :

   ```
   /*Added by CCII for FC driver*/
   #define INCLUDE_CCFC_END            /* CCII FC END DRIVER */

   /* CCII Modification */
   #ifdef INCLUDE_CCFC_END
   #ifndef INCLUDE_PCI
   #define INCLUDE_PCI
   #endif

   #ifdef INCLUDE_BSD
   #undef INCLUDE_BSD
   #endif

   #ifndef INCLUDE_END
   #define INCLUDE_END
   #endif
   #endif

   /*End CCII Modification */
   ```

4. In config.h, change the DEFAULT_BOOT_LINE to use the ccfc driver.

5. Add the folowing to configNet.h.

   (Before the start of the endDevTbl[] decleration.)

   ```
   IMPORT END_OBJ * ccfcLoad (char *initString);
   ```

   (Add the following segment to the endDevTbl[] , before the default last entry.)

   ```
   #ifdef INCLUDE_CCFC_END
       { 0, ccfcLoad, "", NULL, NULL, FALSE},
   #endif /* INCLUDE_CCFC_END */
   ```

6. Rebuild all VxWorks images.

---

[1] *xxx*=X86 for Intel X86 PC, DY4PPC for DY4179/181/182 or 5100 for MV5100 PPC

[2] Refer to Section 3.2

4.2 Loading the Driver Separately

From the VxWorks shell type :

```
ld < ccfcEndxxx.a[1]
```

4.3 Starting the Driver

The driver is started and attached to the MUX with the VxWorks muxDevLoad and muxDevStart commands. The syntax is as follows :

```
muxDevStart ( muxDevLoad ( unitno, ccfcLoad,"", 0, 0 ) )
```

where unitno corresponds to Function[0] = 0 and Function[1] = 1. This is only necessary if the driver was not included in the configNet.h file as described in Section 4.1.

Type `muxShow` to see if the driver was installed.

4.3.1 Command Line Passing of FC Settings

The FC END VxWorks Software Driver supports the changing of some default parameters at startup.

e.g.`muxDevStart (muxDevLoad ( unitno, ccfcLoad,"unitno:NumberOfClusters:MTU",0,0))`

where :

- `unitno` : corresponds to Function[0] = 0 and Function[1] = 1
- `NumberOfClusters` : Number of TX and RX clusters or buckets - default 227 clusters (RX = 100 and TX = 127 clusters), minimum 10 clusters and maximum 254 clusters.
- `MTU` : Max Transfer Unit size in Decimal Bytes - default 32 kBytes - 4 Bytes, minimum 1 kByte and maximum 64 kBytes.

If a parameter is not used a placeholder of 0 **must** be used. This is only applicable for `NumberOfClusters` and `MTU`.

e.g. To change only `NumberOfClusters` to 200 :

```
muxDevStart (muxDevLoad (0,ccfcLoad,"0:200:0",0,0))
```

All parameters preceding the ones you want to change must be specified.

To keep all settings at default values use :

```
muxDevStart ( muxDevLoad ( unitno, ccfcLoad, "", 0, 0 ))
```

4.4 Attaching Driver to TCP/IP stack

Attach the Fibre Channel protocol to the TCP/IP protocol stack with :

```
ipAttach unitno, "ccfc"
```

Use `muxShow` and `ifShow` to confirm.

[1] *xxx*=X86 for Intel X86 PC, DY4PPC for DY4179/181/182 or 5100 for MV5100 PPC

4.5 Loading the FDDI Shim Module

Note : As mentioned in Section 3.2, this module is only necessary in some instances.

Load the FDDI shim with `ld < ccFddiShim.o`.

5. Contact Details

5.1 Contact Person

Direct all correspondence and / or support queries to the Project Manager at C²I² Systems.

5.2 Physical Address

C²I² Systems
Unit 3, Rosmead Place, Rosmead Centre
67 Rosmead Avenue
Kenilworth
Cape Town
7708
South Africa

5.3 Postal Address

C²I² Systems
P.O. Box 171
Rondebosch
7701
South Africa

5.4 Voice and Electronic Contacts

Tel	:	(+27) (0)21 683 5490
Fax	:	(+27) (0)21 683 5435
Email	:	info@ccii.co.za
Email	:	support@ccii.co.za
URL	:	http://www.ccii.co.za/

5.5 Product Support

Support on C²I² Systems products is available telephonically between Monday and Friday from 09:00 to 17:00 CAT. Central African Time (CAT = GMT + 2).

Annexure A

FDDI vs FC

FDDI	FC
ccFddiEndxxx.a	ccfcEndxxx.a
fddiLoad	ccfcLoad
-	ccfcShim.a
*ccfddiGetStats(struct cc_fddi_mib_type *data)* - returns FDDI Management Information Base (MIB)	*ccfddiGetStats(struct cc_fddi_mib_type *data)* - returns a structure with fields set to NULL
ccfddiClrStats(void) - reset counters in the FDDI MIB	*ccfddiClrStats(void)* - reset some fields related to FC
struct cc_fddi_mib_type - all fields related to FDDI MIB	*struct cc_fddi_mib_type* - all fields set to NULL except : rx_packets;0 tx_packets; rx_errors; tx_errors;

Annexure B

Comparing Use of the FC and FDDI END VxWorks Software Drivers

Getting Started With the FC END VxWorks Software Driver	**Getting Started With the FDDI END VxWorks Software Driver**
1. Boot VxWorks and load the ccfcEnd*xxx*.a[1] lib with '`ld < ccfcEndxxx.a`[1]'.	1. Boot VxWorks and load the ccFddiEnd*xxx*.a[1] lib with '`ld < ccfddiEndxxx.a`[1]'.
2. Type '`muxDevStart ( muxDevLoad ( unitno, ccfcLoad,"",0,0 ))`' to attach and start the driver. Verify by using `muxShow`. The *unitno* should be 0 for Function[0] and 1 for Funtion[1] on the FC Adapter.	2. Type '`muxDevStart ( muxDevLoad ( unitno, fddiLoad,"",0,0 ))`' to attach and start the driver. Verify by using `muxShow`. The *unitno* should be 0 for first adapter and 1 for second FDDI Adapter.
3. Attach the Driver to the TCP/IP protocol stack. Type '`ipAttach unitno, "ccfc"' Use muxShow and ifShow` to confirm that the ccfc*unitno* exists and was attached to TCP/IP.	3. Attach the Driver to the TCP/IP protocol stack. Type '`ipAttach unitno, "fddi"' Use muxShow and ifShow` to confirm that the ccfc*unitno* exists and was attached to TCP/IP.
3. Set the IP address using '`ifAddrSet`'. Refer to VxWorks Programmers Guide paragraph '*5.2.5 TCP/IP Internet Protocols and Addresses'* and paragraph '*5.3 Configuring the Network*' for setting up IP host names and IP routing.	3. Set the IP address using '`ifAddrSet`'. Refer to VxWorks Programmers Guide paragraph '*5.2.5 TCP/IP Internet Protocols and Addresses'* and paragraph *'5.3 Configuring the Network*' for setting up IP host names and IP routing.
4. Confirm the FC LAN connection using ping. An example is shown on the next page. Note : This example assumes there is already a NIC on the FC LAN which has been set up with an IP address of 10.0.0.1.	4. Confirm the FDDI LAN connection using ping. An example is shown on the next page. Note : This example assumes there is already a NIC on the FDDI LAN which has been set up with an IP address of 10.0.0.1.

[1] *xxx*=X86 for Intel X86 PC, DY4PPC for DY4179/181/182 or 5100 for MV5100 PPC

```
-> muxDevStart ( muxDevLoad ( 0,ccfcLoad,
"", 0, 0 ))

value = 0 = 0x0

-> ipAttach 0 , "ccfc"
   value = 0 = 0x0


-> ifShow
      ppp (unit number 0):
        Flags: (0x71) UP POINT-TO-POINT
        ARP RUNNING
        Internet address: 172.16.0.2
        Destination Internet address:
        172.16.0.1
        Netmask 0xffff0000 Subnetmask
        0xffff0000
        Metric is 0
        Maximum Transfer Unit size is
        1 500
        5 packets received; 5 packets
        sent
        0 input errors; 0 output errors
        0 collisions
      lo (unit number 0):
        Flags: (0x69) UP LOOPBACK ARP
        RUNNING
        Internet address: 127.0.0.1
        Netmask 0xff000000 Subnetmask
        0xff000000
        Metric is 0
        Maximum Transfer Unit size is
        4 096
        0 packets received; 0 packets
        sent
        0 input errors; 0 output errors
        0 collisions
      ccfc (unit number 0):
        Flags: (0x63) UP BROADCAST ARP
        RUNNING
        Netmask 0xffffffff Subnetmask
        0xffffffff
        Ethernet address is
        00:50:C2:18:D0:00
        Metric is 0
        Maximum Transfer Unit size is
        16 384
        18 packets received; 0 packets
        sent
        0 input errors; 0 output errors
        0 collisions
      value = 18 = 0x12


-> ifAddrSet "ccfc0","10.0.0.4"
      value = 0 = 0x0
```

```
-> muxDevStart ( muxDevLoad ( 0,fddiLoad,
"", 0, 0 ))

value = 0 = 0x0

-> ipAttach 0 , "fddi"
   value = 0 = 0x0


-> ifShow
      ppp (unit number 0):
        Flags: (0x71) UP POINT-TO-POINT
        ARP RUNNING
        Internet address: 172.16.0.2
        Destination Internet address:
        172.16.0.1
        Netmask 0xffff0000 Subnetmask
        0xffff0000
        Metric is 0
        Maximum Transfer Unit size is
        1 500
        5 packets received; 5 packets
        sent
        0 input errors; 0 output errors
        0 collisions
      lo (unit number 0):
        Flags: (0x69) UP LOOPBACK ARP
        RUNNING
        Internet address: 127.0.0.1
        Netmask 0xff000000 Subnetmask
        0xff000000
        Metric is 0
        Maximum Transfer Unit size is
        4 096
        0 packets received; 0 packets
        sent
        0 input errors; 0 output errors
        0 collisions
      fddi (unit number 0):
        Flags: (0x63) UP BROADCAST ARP
        RUNNING
        Netmask 0xffffffff Subnetmask
        0xffffffff
        Ethernet address is
        00:00:5a:45:f0:46
        Metric is 0
        Maximum Transfer Unit size is
        4 491
        18 packets received; 0 packets
        sent
        0 input errors; 0 output errors
        0 collisions
      value = 18 = 0x12


-> ifAddrSet "fddi0","10.0.0.4"
      value = 0 = 0x0
```

```
-> ping "10.0.0.1"
      PING 10.0.0.1: 56 Data Bytes
      64 Bytes from 10.0.0.1: icmp_seq=1.
      time=0. ms
      64 Bytes from 10.0.0.1: icmp_seq=2.
      time=0. ms
      64 Bytes from 10.0.0.1: icmp_seq=3.
      time=0. ms
      64 Bytes from 10.0.0.1: icmp_seq=4.
      time=0. ms
      64 Bytes from 10.0.0.1: icmp_seq=5.
      time=0. ms
```


```
-> ping "10.0.0.1"
      PING 10.0.0.1: 56 Data Bytes
      64 Bytes from 10.0.0.1: icmp_seq=1.
      time=0. ms
      64 Bytes from 10.0.0.1: icmp_seq=2.
      time=0. ms
      64 Bytes from 10.0.0.1: icmp_seq=3.
      time=0. ms
      64 Bytes from 10.0.0.1: icmp_seq=4.
      time=0. ms
      64 Bytes from 10.0.0.1: icmp_seq=5.
      time=0. ms
```


Load the FDDI Shim

In some instances, the FC Adapter will be used to replace the FDDI Adapter in an existing system. In this case, there is no need to change the user applications. The only difference between the FC and FDDI END VxWorks Software Drivers are that the fddiGetStats() routine is replaced by the ccfcGetStats() routine. The shim is present to restore the difference. Figure 1 shows a graphical layout of the difference between the FDDI END VxWorks Software Driver and the FC END VxWorks Software Driver. Users of the FDDI END VxWorks Software Driver will be familiar with the structure **cc_fddi_mib_type** and the following routines :

- **void ccfddiGetStats (struct cc_fddi_mib_type *data)**
- **void ccfddiClrStats(void)**

Although the FC END VxWorks Software Driver is not compatible with the FDDI MIB, calls to the MIB routines will not result in an error as long as the FDDI shim module is loaded. In the FDDI structure **cc_fddi_mib_type**, some fields may or may not correspond with fields in structures of the FC END VxWorks Software Driver. As per default, all the fields will be set to NULL.

Load the FDDI shim with `'ld < ccFddiShim.o'.`