C²I² Systems

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

Communications
Computer Intelligence
Integration

User Manual

for the

FDDI Adapter

4.3 BSD VxWorks Software Driver

C²I² Systems Document No.	CCII/FDDI/6-MAN/002
Document Issue	1.2
Issue Date	2009-08-20
Print Date	2009-08-20
File Name	P:\FDDI\TECH\MAN\CFDMAN02.WPD
Distribution List No.	

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 version created for Dy4 4.3 BSD Driver review.	1998-09-15	-
0.2	Update prior to release to Dy4.	1998-10-15	-
1.0	Moved version references to release notes.	1998-11-26	-
1.1	Added chapter : ‘Application Program Interface (API)’.	1998-12-24	CCII/FDDI/6-ECP/003
1.2	Improve document naming consistency.	2009-08-20	CCII/FDDI/6-/ECP/034

Contents

Abbreviations and Acronyms

BIT	Built-in Test
BSD	Berkley Sockets Devices
BSP	Board Support Package
FDDI	Fibre Distributed Data Interface
ICMP	Internet Control Message Protocol
IP	Internet Protocol
LAN	Local Area Network
MIB	Management Information Base
NIC	Network Interface Card
PCI	Peripheral Component Interconnect
PMC	Peripheral Component Interconnect Mezzanine Card
SMT	Station Management
TCP	Transmission Control Protocol
VME	Versa Module Eurocard

1. Scope

1.1 Identification

This document is the user manual for the C²I² Systems Fibre Distributed Data Interface (FDDI) Adapter 4.3 Berkley Sockets Devices (BSD) VxWorks Software Driver.

1.2 Introduction

The FDDI 4.3 BSD VxWorks Software Driver was developed specifically to operate on Dy4's SVME178 and Radstone's PPC1a host carrier cards. As such the drivers binaries are provided with explicit installation instructions. The FDDI 4.3 BSD VxWorks Software Driver conforms to the VxWorks 4.3 BSD driver model and the driver will interface to VxWorks using the standard.

The FDDI adapters attach computers to 100 Mbit/s FDDI networks using fibre optic cable.

The driver software distribution consists of the following files :

ccFddi.a	FDDI driver object file.
readme.txt	Installation notes.
Release.txt	Release notes and revision history : Please check this file for information on the latest updates.
ccMib.h	Defines the structure `cc_fddi_mib_type` for accessing the FDDI Management Information Base (MIB).
ShowStat.c	Sample C program for accessing the FDDI MIB.

2. Applicable and Reference Documents

2.1 Applicable Documents

2.1.1 DI-IPSC-81443, *Data Item Description for a Software User Manual*.

2.1.2 ISO/IEC 9314-6 / X3.229, *Information technology -- Fibre Distributed Data Interface (FDDI) -- Part 6: Station Management (SMT)*, dated 7 October 1993.

2.1.3 CCII/A500/IMS/6-MAN/1, *PMC FDDI Network Card Installation Guide*.

2.1.4 *VxWorks 5.4 Programmer's Guide*, edition 1.

2.1.5 RFC-1512, *FDDI Management Information Base*, dated September 1993.

2.2 Reference Documents

2.2.1 ANSI INCITS 229, *Information Systems - Fibre Distributed Data interface (FDDI) - Station Management (SMT)*, dated 1994-01-01.

3. Installation Procedure

This paragraph describes the installation instructions for the PMC FDDI 4.3 BSD VxWorks Driver.

3.1 To Build the FDDI 4.3 BSD VxWorks Software Driver into the VxWorks Kernel

1. Copy ccFddi.a to your Board Support Package (BSP) library directory (/tornado/target/config/svme178/lib).

2. Edit the Makefile in the BSP directory (/tornado/target/config/svme178).

   Find the line
   ```
   MACH_EXTRA  =
   ```

   and replace with
   ```
   MACH_EXTRA  =     ./lib/ccFddi.a
   ```

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

   (before "`#define DEFAULT_BOOT_LINE`") :

   ```
   #define INCLUDE_ FDDI
   #ifdef INCLUDE_FDDI
   #define NETIF_USR_DECL IMPORT int ccfddiattach () ;
   #define NETIF_USR_ENTRIES \
        {     "fddi", \
        ccfddiattach, \
        /* Unit = */ 0, \
        /* Receive buffers = */ 0 /* = use default */, \
        /* Transmit buffers = */ 0 /* = use default */ \
        },
   #endif /* INCLUDE_FDDI */
   ```

4. Rebuild all VxWorks images.

3.2 To Load the Driver Software Separately

From the VxWorks shell, type :

```
ld < ccFddi.a
```

3.3 Starting the Driver

The driver is started with the ccfddiattach command. The syntax is as follows :

```
ccfddiattach (0, 0, 0)
```

4. Application Program Interface (API)

4.1 Driver Functions

4.1.1 int ccfddiattach(int unit, int n_rx_buffers, int n_tx_buffers)

This function makes the FDDI adapter available to the operating system as a network interface.

`unit` : Must be 0 (only one adapter is supported).

`n_rx_buffers` : Specifies the number of receive buffers to allocate. This is constrained to be in the range 10 - 100. If 0 is specified, a default value of 13 will be used.

`n_tx_buffers` : Specifies the number of transmit buffers to allocate. This is constrained to be in the range 10 - 100. If 0 is specified, a default value of 20 will be used.

Warning : Changing any of these parameters from default values may have a profound effect on driver performance. Increasing the number of allocated buffers may worsen driver performance in some situations.

The function returns TRUE if initialisation is unsuccessful.

4.1.2 void ccfddiSuspendDriver()

This function allows the driver to be temporarily suspended. After calling this function, the FDDI network interface will be unavailable and will be marked as 'down'. Note that the network card is reset by this operation, and that any packets being transmitted or received at this time will be lost.

This function is called by the Built-in Test (BIT) Software when it needs exclusive access to the adapter.

4.1.3 void ccfddiResumeDriver()

This function allows the driver to resume operation after being suspended by `ccfddiSuspendDriver`. Normally this function would only be called by the BIT Software to return control of the adapter to the driver. If the BIT Software terminates abnormally (the user presses control-C for example), this command may be issued to resume normal driver operation.

4.1.4 void ccfddiGetStats(struct cc_fddi_mib_type *data)

This function allows an application to access information about the driver as stored in the MIB.

4.1.5 void ccfddiClrStats(void)

This function resets the following counters in the FDDI MIB :

- fddiMACFrame-Ct
- fddiMACCopied-Ct
- fddiMACTransmit-Ct
- fddiMACError-Ct
- fddiMACLost-Ct
- fddiPORTLCTFail-Ct
- fddiPORTLem-Reject-Ct
- fddiPORTLem-Ct

4.2 Driver Data Structures

4.2.1 cc_fddi_mib_type

```
struct cc_fddi_mib_type
      {
      u32       rx_packets;          /* total packets received */
      u32       tx_packets;          /* total packets transmitted */
      u32       rx_errors;           /* bad packets received*/
      u32       tx_errors;           /* packet transmit problems  */
      u32       rx_dropped;          /* no space in linux buffers */
      u32       tx_dropped;          /* no space available in VxWorks */
      u32       multicast;           /* multicast packets received */
      u32       transmit_collision;  /* always 0 for FDDI */

      /* Detailed FDDI statistics. */
      u8        smt_station_id [8];
      u32       smt_op_version_id;
      u32       smt_hi_version_id;
      u32       smt_lo_version_id;
      u8        smt_user_data 32];
      u32       smt_mib_version_id;
      u32       smt_mac_cts;
      u32       smt_non_master_cts;
      u32       smt_master_cts;
      u32       smt_available_paths;
      u32       smt_config_capabilities;
      u32       smt_config_policy;
      u32       smt_connection_policy;
      u32       smt_t_notify;
      u32       smt_stat_rpt_policy;
      u32       smt_trace_max_expiration;
      u32       smt_bypass_present;
      u32       smt_ecm_state;
      u32       smt_cf_state;
      u32       smt_remote_disconnect_flag;
      u32       smt_station_status;
      u32       smt_peer_wrap_flag;
      u64       smt_time_stamp;
      u64       smt_transition_time_stamp;
      u32       mac_frame_status_functions;
      u32       mac_t_max_capability;
      u32       mac_tvx_capability
      u32       mac_available_paths;
      u32       mac_current_path;
      u8        mac_upstream_nbr[CC_ALEN];
      u8        mac_downstream_nbr[CC_ALEN];
      u8        mac_old_upstream_nbr[CC_ALEN];
      u8        mac_old_downstream_nbr[CC_ALEN];
      u32       mac_dup_address_test;
      u32       mac_requested_paths;
      u32       mac_downstream_port_type;
      u8        mac_smt_address[CC_ALEN];
      u32       mac_t_req;
      u32       mac_t_neg;
      u32       mac_t_max;
      u32       mac_tvx_value;
      u32       mac_frame_cts;
      u32       mac_copied_cts;
      u32       mac_transmit_cts;
      u32       mac_error_cts;
      u32       mac_lost_cts;
      u32       mac_frame_error_threshold;
      u32       mac_frame_error_ratio;
```

```
u32       mac_rmt_state;
u32       mac_da_flag;
u32       mac_una_da_flag;
u32       mac_frame_error_flag;
u32       mac_ma_unitdata_available;
u32       mac_hardware_present;
u32       mac_ma_unitdata_enable;
u32       path_tvx_lower_bound;
u32       path_t_max_lower_bound;
u32       path_max_t_req;
u32       path_configuration [8];
u32       port_my_type [2];
u32       port_neighbor_type [2];
u32       port_connection_policies [2];
u32       port_mac_indicated [2];
u32       port_current_path [2];
u8        port_requested_paths [3*2];
u32       port_mac_placement [2];
u32       port_available_paths [2];
u32       port_pmd_class [2];
u32       port_connection_capabilities [2];
u32       port_bs_flag [2];
u32       port_lct_fail_cts [2];
u32       port_ler_estimate [2];
u32       port_lem_reject_cts [2];
u32       port_lem_cts [2];
u32       port_ler_cutoff [2];
u32       port_ler_alarm [2];
u32       port_connect_state [2];
u32       port_pcm_state [2];
u32       port_pc_withhold [2];
u32       port_ler_flag [2];
u32       port_hardware_present [2];
};
```

This structure, defined in `ccMib.h`, is used to return the results of `ccfddiGetStats`. A detailed explanation of the meanings of these fields may be found in the ANSI document *FDDI Station Management (SMT).* Note that these definitions differ slightly from those defined in RFC-1512 (*FDDI Management Information Base).*

The types `u8, u32` and `u64` are unsigned 8-bit, 32-bit, and 64-bit integers respectively. These may be defined in Vxworks for PowerPC as `unsigned char, unsigned int` and `unsigned long int` respectively.

4.3 Application Example

Included in the driver distribution is sample source code (`ShowStat.c`) which shows how to use the `ccfddiGetStats` function to retrieve the FDDI MIB information.

The program simply prints out the contents of the MIB. Included in the source code are two functions, `ccfddiTime2sec` and `ccfddiTimerTwosComp2sec`, which demonstrate how ANSI time values are interpreted.

5. Getting Started

1. Boot VxWorks.

2. Type ‘ccfddiattach 0,0,0' to attach the driver. Verify that the fddi0 device exists by typing ‘ifShow’.

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 FDDI connection using ping. An example is shown below.

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

```
-> ccfddiattach 0,0,0
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
```

6. Contact Details

6.1 Contact Person

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

6.2 Physical Address

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

6.3 Postal Address

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

6.4 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/

6.5 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).