C²I² Systems

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

Communications
Computer Intelligence
Integration

Generic User Manual

for the

FDDI Adapter

VxWorks BIT Application

C²I² Systems Document No.	CCII/FDDI/6-MAN/003
Document Issue	1.2
Issue Date	2009-08-20
Print Date	2009-08-20
File Name	P:\FDDI\TECH\MAN\CFDMAN03.WPD
Distribution List No.	DN 0090

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	2009-08-21
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.
1.0	Initial version based on Dy4 document.	2000-11-28	-
1.1	Correct documentation errors in Figure 2 and in Section 5.7.	2002-10-22	
1.2	Improve document naming consistency.	2009-08-20	CCII/FDDI/6-ECP/035

Contents

Abbreviations and Acronyms

API	Application Program Interface
ASIC	Application Specific Integrated Circuit
BIT	Built-in Test
BIST	Built-in Self Test
BIU	Bus Interface Unit
BMU	Buffer Management Unit
BSP	Board Support Package
CLS	Cache Line Size
DAS	Dual Attached Station
DMA	Direct Memory Access
FDDI	Fibre Distributed Data Interface
FORMAC	Fibre Optic Ring Media Access Controller
FPROM	Flash Programmable Read-only Memory
HCC	Host Carrier Card
HPI	Host Processor Interface
IRQ	Interrupt Request
LED	Light Emitting Diode
MAC	Media Access Control
Mbit/s	Megabits per Second
MDR	Memory Data Register
MWI	Memory Write and Invalidate
OBS	Optical Bypass Switch
PCI	Peripheral Component Interconnect
PDR	Physical Data Receiver
PDT	Physical Data Transmitter
PLC	Physical Layer Controller
PLC-S	Physical Layer Controller with Scrambler
PMC	Peripheral Component Interconnect Mezzanine Card
PROM	Programmable Read-only Memory
RAM	Random Access Memory
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 VxWorks Built-in Test (BIT) Application.

1.2 System Overview

The FDDI VxWorks BIT Application was developed specifically to be ported to operate on a variety of Host Carrier Cards (HCCs)[1]. As such the FDDI VxWorks BIT Application binaries are provided with explicit installation instructions. The FDDI VxWorks BIT Application tests the low-level integrity of the FDDI hardware.

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

The BIT Application consists of the following files :

ccFdBit.a	FDDI BIT object file.
ccFdBit.h	Header file for user applications.
ccFddiX.o	Stub to allow BIT Application to be linked into kernel without FDDI driver.
BitDemo.c	Sample BIT Application.
Release.txt	Release notes and revision history: Please check this file for information on the latest updates.

[1] Currently supports Dy4 SVME178, Radstone PPC, Power 4B and MVME 5100.

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 CCII/FDDI/6-MAN/008, *Installation Guide for the FDDI Adapter*.

2.1.3 *VxWorks 5.3.1 Programmer's Guide*, edition 1.

2.2 Reference Documents

None.

3. Installation Procedure

This paragraph describes the installation instructions for the FDDI VxWorks BIT Application.

3.1 To Build the BIT Application into the VxWorks Kernel

1. Copy ccFddiX.o and ccFdBit.a to your Berkley Sockets Devices (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/ccFddiX.o ./lib/ccFdBit.a
   ```

3. Rebuild all VxWorks images.

3.2 To Build the FDDI Software Driver and BIT Application into the VxWorks Kernel

1. Copy ccFdBit.a to your BSP library directory (/tornado/target/config/svme178/lib).

2. Copy ccFddi.a (from your driver distribution) to your BSP library directory (/tornado/target/config/svme178/lib).

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

   Find the line
   ```
   MACH_EXTRA =
   ```

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

4. 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 */
   ```

5. Rebuild all VxWorks images.

3.3 To Load the BIT Application Separately

From the VxWorks shell, type :

```
ld < ccFdBit.a
```

The BIT Application may be operated without the FDDI Software Driver installed. Depending on the release version in use, the following message might be displayed on loading the BIT Application :

```
Undefined symbols:
ccfddiResumeDriver
ccfddiSuspendDriver
Warning: object module may not be usable because of undefined symbols.
```

This is normal and will not affect operation of the software.

3.4 Platform Specific Variations

This manual assumes that the executable is named ccFdBit.a. In order to prevent confusion, platform specific variations are released with a unique filename, as listed below. The user should simply rename this file to ccFdBit.a before using it, although the platform specific filename may be used as is, if desired. This manual however, refers to the file as ccFdBit.a.

Dy4 178	ccFdBitSVME178v*xxxx*.a
RAD PPC	ccFdBitRadPPCv*xxxx*.a
POWER 4B	ccFdBitP4Bv*xxxx*.a
MVME5100	ccFdBitMVME5100v*xxxx*.a

Where *xxxx* represents the current version number of the release.

4. Application Program Interface (API)

All function prototypes, data types and structures described in this section are defined in the header file `ccFdBit.h`.

4.1 BIT Functions

4.1.1 int ccfddiInitBIT (cc_test_message_style_type init_messages_style)

This function is used to initialise the BIT Application. The argument specifies the amount of text displayed during initialisation of the BIT Application.

The function returns TRUE if initialisation is unsuccessful.

4.1.2 int ccfddiPOST (cc_tests_type *post_tests, cc_tests_report_type *post_results)

This function is used to initialise the BIT Application and run a set of startup tests.

If `post_tests` is NULL, a default set of tests appropriate to startup will be executed (i.e. no loopback or Light Emitting Diode (LED) tests). The structure that defines this is called `cc_default_post_tests`.

If `post_results` is NULL, results will be written to the location `cc_default_post_results`.

The function returns TRUE if any of the tests return an error.

4.1.3 int ccfddiTest (cc_tests_type *tests, cc_tests_report_type *results)

This function runs a set of tests as specified in `tests` and places the results in `results`.

The function returns TRUE if any of the tests return an error.

4.1.4 int ccfddiShowTestResults (cc_tests_report_type *results)

This function displays the test results in `results` as shown in the following example :

```
PROM Check                    PASSED
Timer and Hardware IRQ        PASSED
FORMAC Register Check         PASSED
RAM Check over MDR            PASSED
RAM Check over DMA Engine     PASSED
Memory Data Transfer Rate     PASSED
PLC 1 Test                    PASSED
PLC 2 Test                    FAILED (Code 550)
Bypass Test                   SKIPPED
Wrap Around Check             PASSED
FORMAC Loopback Check         FAILED (Code 191)
Send and Receive Long Frames  PASSED
Send and Master Access Loop   FAILED (Code 191)
FORMAC Ring-Op Status         PASSED
Configuration Registers Check PASSED
Special Card Check            PASSED
ASIC Check                    PASSED
LED Test                      PASSED
```

The function returns TRUE if any of the tests reported a failure.

4.2 BIT Data Types

4.2.1 cc_test_message_style_type

```
typedef enum {cc_quiet = 0,
              cc_dots = 1,
          cc_detailed = 2}
cc_test_message_style_type;
```

This type is used in the `cc_test_option_type` structure to control the amount of text displayed during execution of a particular test.

The `cc_quiet` option is used when no output is desired.

The `cc_dots` option prints a series of dots to indicate test progress.

The `cc_detailed` option displays information which may be helpful in diagnosing faults.

4.2.2 cc_run_status_type

```
typedef enum {cc_skipped = 0,
              cc_failed  = 1,
              cc_passed  = 2}
         cc_run_status_type;
```

This type is used in the `cc_test_result_type` structure to indicate the outcome of a particular test.

4.3 BIT Data Structures

4.3.1 cc_test_option_type

```
typedef struct
    {
    int do_test;          /* Set this to run test,
                           * otherwise test is skipped */
    int stop_on_error;    /* If this is set, no more tests
                           * will be run if there
                           * is an fail */
    int loop_forever;     /* Tests with this bit set will
                           * be run continuously */
    cc_test_message_style_type test_message_style;
                          /* Determines appearance of test
                           * messages */
    }
    cc_test_option_type;
```

This structure controls the execution of a particular test.

If the `do_test` flag is set to TRUE, the test executed, otherwise it is skipped.

If the `stop_on_error` flag is set to TRUE for a test, and the test returns an error, then testing terminates immediately (no further tests are run).

If the `loop_forever` flag is set to TRUE for any test, then testing continues indefinitely (i.e. all tests with this flag set will be repeated continually). The `stop_on_error` flag would normally be used in conjunction with this. To abort continuous testing, use the Control-C interrupt mechanism. If the driver was suspended for testing, then use `ccfddiResumeDriver` to resume normal driver operation.

The `cc_test_message_style` determines how much output is displayed to the screen as described in paragraph 4.2.1.

4.3.2 cc_test_result_type

```
typedef struct
    {
    cc_run_status_type run_status;
    int                result_code;
    }
    cc_test_result_type;
```

This structure captures the outcome of a particular test.

Test result codes are listed in Section 6.

4.3.3 cc_tests_type

```
typedef struct
    {
    int                          loopback_mode;
    cc_test_message_style_type   init_messages_style;
    cc_test_option_type          prom_check;
    cc_test_option_type          timer_and_hardware_irq;
    cc_test_option_type          formac_register;
    cc_test_option_type          ram_check_over_mdr;
    cc_test_option_type          ram_over_dma_engine;
    cc_test_option_type          memory_data_transfer_rate;
    cc_test_option_type          plc_1_test;
    cc_test_option_type          plc_2_test;
    cc_test_option_type          bypass_test;
    cc_test_option_type          wrap_around_check;
    cc_test_option_type          formac_loopback;
    cc_test_option_type          send_and_receive_long_frames;
    cc_test_option_type          send_and_master_access_loop;
    cc_test_option_type          formac_ring_op_status;
    cc_test_option_type          configuration_registers_check;
    cc_test_option_type          special_card_check;
    cc_test_option_type          asic_check;
    cc_test_option_type          led_test;
    }
    cc_tests_type;
```

This structure controls the execution of all the tests.

If `loopback_mode` is TRUE, then the user must install wrap plugs on both ports before testing, as shown in Figure 1 :

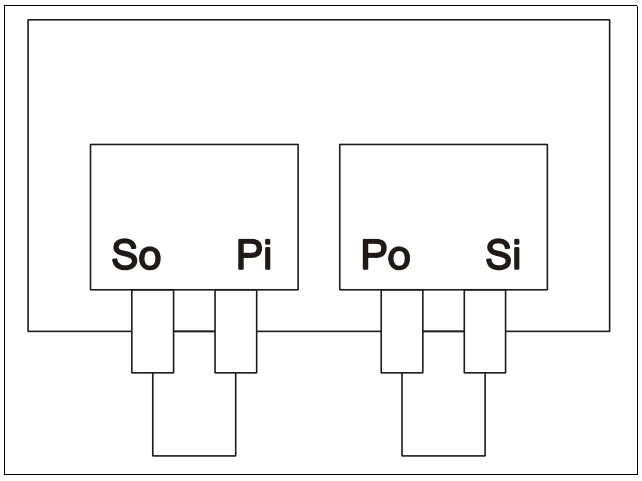

Figure 1 : Loopback Wrap Plugs

The `loopback_mode` flag affects the following tests :

1. PLC 1 Test
2. PLC 2 Test
3. FORMAC Loopback
4. Send and Receive Long Frames
5. Send and Master Access Loop

The `init_messages_style` member specifies the amount of text displayed during initialisation of the BIT Application when calling `ccfddiPOST`.

```
cc_tests_report_type
typedef struct
    {
    cc_test_result_type    prom_check;
    cc_test_result_type    timer_and_hardware_irq;
    cc_test_result_type    formac_register;
    cc_test_result_type    ram_check_over_mdr;
    cc_test_result_type    ram_over_dma_engine;
    cc_test_result_type    memory_data_transfer_rate;
    cc_test_result_type    plc_1_test;
    cc_test_result_type    plc_2_test;
    cc_test_result_type    bypass_test;
    cc_test_result_type    wrap_around_check;
    cc_test_result_type    formac_loopback;
    cc_test_result_type    send_and_receive_long_frames;
    cc_test_result_type    send_and_master_access_loop;
    cc_test_result_type    formac_ring_op_status;
    cc_test_result_type    configuration_registers_check;
    cc_test_result_type    special_card_check;
    cc_test_result_type    asic_check;
    cc_test_result_type    led_test;
    }
    cc_tests_report_type;
```

This structure captures the outcome of all tests.

4.3.4 Application Example

The following source file is a simple demonstration of how to execute the BIT functions :

```
#include "ccFdBit.h"

cc_tests_type tests =
       {FALSE, /* Loopback NOT connected */
       cc_detailed,        /* {do_test, stop_on_error, loop_forever, test_message_style} */
       {TRUE, FALSE, FALSE, cc_detailed}, /* prom_check */
       {TRUE, FALSE, FALSE, cc_detailed}, /* timer_and_hardware_irq */
       {TRUE, FALSE, FALSE, cc_detailed}, /* formac_register */
       {TRUE, FALSE, FALSE, cc_detailed}, /* ram_check_over_mdr */
       {TRUE, FALSE, FALSE, cc_detailed}, /* ram_over_dma_engine */
       {TRUE, FALSE, FALSE, cc_detailed}, /* memory_data_transfer_rate */
       {TRUE, FALSE, FALSE, cc_dots},      /* plc_1_test */
       {TRUE, FALSE, FALSE, cc_dots},      /* plc_2_test */
       {FALSE, FALSE, FALSE, cc_detailed},/* bypass_test */
       {TRUE, FALSE, FALSE, cc_detailed}, /* wrap_around_check */
       {TRUE, FALSE, FALSE, cc_detailed}, /* formac_loopback */
       {TRUE, FALSE, FALSE, cc_detailed}, /* send_and_receive_long_frames */
       {TRUE, FALSE, FALSE, cc_detailed}, /* send_and_master_access_loop */
       {TRUE, FALSE, FALSE, cc_detailed}, /* formac_ring-op_status */
       {TRUE, FALSE, FALSE, cc_detailed}, /* configuration_registers_check */
       {TRUE, FALSE, FALSE, cc_quiet},     /* special_card_check */
       {TRUE, FALSE, FALSE, cc_quiet},     /* asic_check */
       {TRUE, FALSE, FALSE, cc_quiet},     /* led_test */
       };
cc_tests_report_type results;

int BitDemo (void)
```

```
{
      ccfddiInitBIT (cc_detailed);
      ccfddiTest (&tests, &results);
      return ccfddiShowTestResults (&results);
}
```

5. Test Descriptions

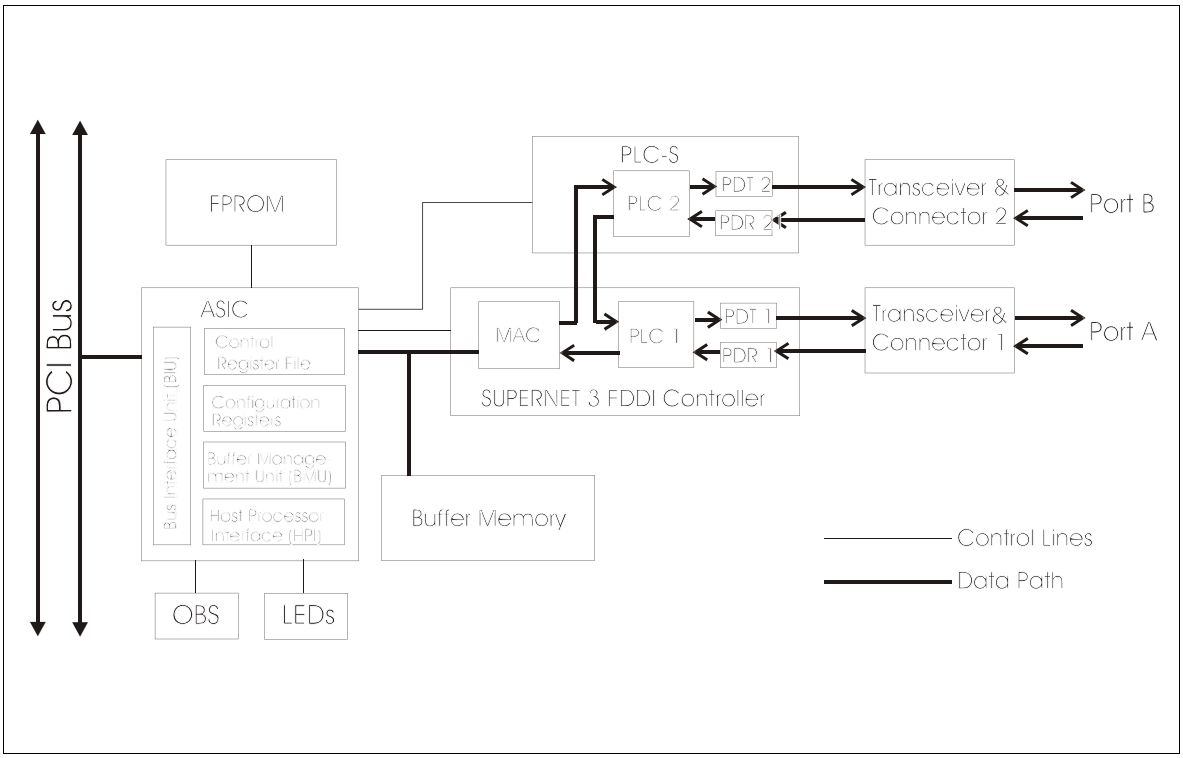

Figure 2 : FDDI Adapter Block Diagram

5.1 PROM Check

5.1.1 Description

This reads the contents of the Programmable Read-only Memory (PROM) to an array, and does a compare against the first three bytes of the Media Access Control (MAC) address. This is done to make sure the MAC address falls within reasonable bounds. The MAC address is displayed if the `cc_detailed` mode is selected.

5.1.2 Test Coverage

This test covers functionality of the block labelled ‘FPROM’ in Figure 2.

5.2 Timer and Hardware Interrupt Request (IRQ)

5.2.1 Description

This tests whether the timer can be used correctly, whether it times out, and whether the timer interrupt occurs correctly.

5.2.2 Test Coverage

This test covers partial functionality of the blocks labelled ‘BIU’ and ‘MAC’ in Figure 2.

5.3 FORMAC Register

5.3.1 Description

This checks the Fibre Optic Ring Media Access Controller (FORMAC) and programmable registers. The test uses all read-only and read-write registers. It also tests the initialisation of counters and other registers. In addition, it executes a test on the Address Filter Function of the SUPERNET 3 chipset.

5.3.2 Test Coverage

The test covers partial functionality of the blocks labelled ‘Control Register File’ and ‘MAC’ in Figure 2.

5.4 RAM Check Over MDR

5.4.1 Description

This reads from and writes to Random Access Memory (RAM) on the FDDI Adapter using the Memory Data Register (MDR), and verifies the data.

5.4.2 Test Coverage

This test covers functionality of the block labelled ‘Buffer Memory’ in Figure 2.

5.5 Random Access Memory Over Direct Memory Access Engine

5.5.1 Description

This reads from and writes to RAM on the FDDI Adapter using Direct Memory Access (DMA) master transfers.

5.5.2 Test Coverage

This test covers functionality of the block labelled ‘Buffer Memory’ in Figure 2.

5.6 Memory Data Transfer Rate

5.6.1 Description

This test measures raw master access speed, and prints the values to the console. The speed is measured with differing Cache Line Size (CLS), byte aligned or misaligned, and in different Memory Write and Invalidate (MWI) modes.

5.6.2 Test Coverage

Although not strictly a test, this will exercise functionality of the Bus Interface Unit (BIU) and Host Processor Interfaces (HPI) of the Application Specific Integrated Circuit (ASIC).

5.7 PLC 1 (Port A) Test

5.7.1 Description

This tests the IRQ, executes the Physical Layer Controller (PLC) Built-in Self Test (BIST) for Port A, does bit signalling loops, initialises timer, checks the read-only and read-write registers. This test requires wrap plugs to be installed as shown in Figure 1.

5.7.2 Test Coverage

This test covers functionality of the block labelled ‘PLC 1’ in Figure 2.

5.8 PLC 2 (Port B) Test

5.8.1 Description

This tests the IRQ, executes the PLC BIST for Port B, does bit signalling loops, initialises timer, checks the read-only and read-write registers. This test requires wrap plugs to be installed as shown in Figure 1.

5.8.2 Test Coverage

This test covers functionality of the block labelled ‘PLC 2’ in Figure 2.

5.9 Bypass Test

5.9.1 Description

This tests whether the optical bypass is present or not. If the optical bypass is present, the user will hear the sound of the switch clicking on and off. This test never returns a failed result.

5.9.2 Test Coverage

This test covers functionality of the block labelled ‘OBS’ in Figure 2.

5.10 Wrap Around Check

5.10.1 Description

This test checks if the FORMAC receives frames correctly when the queues wrap in buffer memory. Short frames (20 Bytes) are sent and received in loopback. It also checks DMA transfer.

5.10.2 Test Coverage

This test covers functionality of the blocks labelled ‘BMU’, ‘BIU’ and ‘MAC’ in Figure 2.

5.11 FORMAC Loopback

5.11.1 Description

This test verifies that the FORMAC can send and receive data without using DMA master access. Various levels of loopback are performed (after MAC, after PLC 1, after PDT/R 1, after Transceiver and Connector 1, after PLC 1 and 2, after PDT/R 2 and after Transceiver and Connector 2. This test requires wrap plugs to be installed as shown in Figure 1.

5.11.2 Test Coverage

This test covers functionality of the following blocks from Figure 2 :

ASIC, SUPERNET 3 FDDI Controller, PLC-S, Transceiver and Connector 1, Transceiver and Connector 2.

5.12 Send and Receive Long Frames

5.12.1 Description

This test sends and receives long frames using the synchronous and asynchronous transmit queues. Long frames (various lengths up to 4 400 Bytes) are sent and received in loopback. DMA transfer is tested as well. This test requires wrap plugs to be installed as shown in Figure 1.

5.12.2 Test Coverage

This test covers functionality of the following blocks from Figure 2 :

ASIC, SUPERNET 3 FDDI Controller, PLC-S, Transceiver and Connector 1.

5.13 Send and Master Access Loop

5.13.1 Description

This test shall prove that the FORMAC can send and receive data using DMA master access. Various levels of loopback are performed (after MAC, after PLC 1, after PDT/R 1, after Transceiver and Connector 1, after PLC 1 and 2, after PDT/R 2 and after Transceiver and Connector 2. This test requires wrap plugs to be installed as shown in Figure 1.

5.13.2 Test Coverage

This test covers functionality of the following blocks from Figure 2 :

ASIC, SUPERNET 3 FDDI Controller, PLC-S, Transceiver and Connector 1, Transceiver and Connector 2.

5.14 FORMAC Ring-Op Status

5.14.1 Description

This test checks that the Ring-Op interrupt occurs in the FORMAC.

5.14.2 Test Coverage

This test covers partial functionality of the blocks labelled ‘BIU’ and ‘MAC’ in Figure 2.

5.15 Configuration Registers Check

5.15.1 Description

This test checks the configuration registers.

5.15.2 Test Coverage

This test covers functionality of the blocks labelled ‘BIU’ and ‘Configuration Registers’ in Figure 2.

5.16 Special Card Check

5.16.1 Description

This runs the parity mode tests, checks all timers in test mode, checks whether writes can be made to I/O space, tests descriptor bits and checks the Bank 0 special registers.

5.16.2 Test Coverage

This test covers functionality of the blocks labelled ‘BIU’, ‘BMU’, ‘HPI’ and part of ‘MAC’ in Figure 2.

5.17 ASIC Check

5.17.1 Description

This checks ASIC functions, as well as the DMA engine with all possible alignments.

5.17.2 Test Coverage

This test covers functionality of the blocks labelled ‘BIU’, ‘BMU’, ‘HPI’ and ‘Buffer Memory’ in Figure 2.

5.18 LED Test

5.18.1 Description

The three adapter LEDs are turned on in all eight combinations, for one second each. The user must verify that these display correctly. This test never returns a failed result.

5.18.2 Test Coverage

This test covers functionality of the blocks labelled ‘BIU’ and ‘LEDs’ in Figure 2.

6. **Test Result Codes**

Code	Description
150	dma_read_check()
160	Mapped POS_106 register defective
161	LED register
162	Cannot switch LED 2 (DAS) on
163	Cannot switch LED 2 (DAS) off
164	Mapped POS_103 register defective (0x36, FM1)
165	Mapped POS_100 register defective
166	Mapped POS_101 register defective
167	Mapped POS_102 register defective
169	Mapped POS_104 register defective
170	Mapped POS_103 register defective (0x2e)
171	CSR_A defective (adapter disabled)
172	CSR_A defective (adapter enabled)
173	CSR_A defective (fifo reset)
174	CSR_A defective (fifo enable)
175	CSR_A defective (IRQ other enabled)
176	CSR_A defective (IRQ other disabled)
177	CSR_A defective (IRQ check enabled)
178	CSR_A defective (IRQ check disabled)
179	CSR_A defective (IRQ RTM enabled)
180	CSR_A defective (IRQ RTM disabled)
181	CSR_A defective (IRQ Terminal count enabled)
182	CSR_A defective (IRQ Terminal count disabled)
183	Cannot switch LED_0 on and LED_1 off
184	Cannot switch LED_0 off and LED_1 on
185	Cannot switch LED_0 off and LED_1 off
186	Cannot switch LED 2 (DAS) on
187	Cannot switch LED 2 (DAS) off
188	Cannot switch to bypass station (DAS)
189	Cannot switch to insert station (DAS)
190	Host request register defective : value cannot be read
D191	MAC does not go to Ring-Op
D191	Host request register defective : should be in read request
192	Host request register defective : should not be in read request
193	Page register defective
199	Timeout ("timer data path\n",1)
FORMAC+ Register Tests	
200	FM_IMSK1U
201	FM_IMSK1L
202	FM_IMSK2U
203	FM_IMSK2L
204	FM_SAID
205	FM_LAIM
206	FM_LAIC

Code	Description
207	FM_LAIL
208	FM_SAGP
209	FM_LAGM
210	FM_LAGC
211	FM_LAGL
220	FM_PRI1
221	FM_PRI2
222	FM_TSYNC
224	FM_FRMTHR
225	FM_EACB
226	FM_EARV
227	FM_EAS
228	FM_EAA0
229	FM_EAA1
230	FM_EAA2
231	FM_SACL
232	FM_SABC
233	FM_WPXSF
234	FM_RPXSF
235	FM_RPR
236	FM_WPR
237	FM_SWPR
238	FM_WPXS
239	FM_WPXA0
240	FM_WPXA1
241	FM_WPXA2
242	FM_SWPXS
243	FM_SWPXA0
244	FM_SWPXA1
245	FM_SWPXA2
246	FM_RPXS
247	FM_RPXA0
248	FM_RPXA1
249	FM_RPXA2
250	FM_MARR
251	FM_MARW
254	FM_FCNTR
255	FM_LCNTR
256	FM_ECNTR
FORMAC+ Register Tests (Initial Values)	
261	FM_MIR1
262	FM_MIR0
263	FM_PRI0
264	FM_PRI1
265	FM_PRI2 (SN2)
268	FM_TNEG

Code	Description
269	FM_TSYNC
270	FM_FCNTR
271	FM_ECNTR
272	FM_LCNTR
273	FM_TMAX
274	FM_THT
274	FM_TVX
274	FM_TMSYNC (SN3)
274	FM_ST1U (SN3)
274	FM_ST1L (SN3)
274	FM_ST2U (SN3)
274	FM_ST2L (SN3)
274	FM_ST3U (SN3)
274	FM_ST3L (SN3)
274	FM_IMSK1U (SN3)
274	FM_IMSK1L (SN3)
274	FM_IMSK2U (SN3)
274	FM_IMSK2L (SN3)
274	FM_IMSK3U (SN3)
274	FM_IMSK3L (SN3)
274	FM_IVR (SN3)
274	FM_IMR (SN3)
274	FM_SAID (SN3)
274	FM_LAIM (SN3)
274	FM_LAIC (SN3)
274	FM_LAIL (SN3)
274	FM_SAGP (SN3)
274	FM_LAGM (SN3)
274	FM_LAGC (SN3)
274	FM_LAGL (SN3)
274	FM_MDREG1 (SN3)
274	FM_MDREG2 (SN3)
274	FM_MDREG3 (SN3)
274	FM_STMCHN (SN3)
274	FM_FSCNTR (SN3)
274	FM_FRMTHR (SN3)
Address Filter Test (IFCP / SUPERNET 3 only)	
280	IFCP_3 Interrupt already set
281	Status register not cleared
282	AF Built-in Self Test timeout!
283	IRQ occured but AF_BIST_DONE (ST3) not set
284	IRQ occured but BIST_DONE (AFSTAT) not set
285	IRQ occured but DONE not set
286	AF BIST signature wrong
302	PROM : wrong MAC address (man-code or IBM 4. Byte)
303	Error on board RAM (RBC)

Code	Description
304	PROM : error in LOGO
307	Error on board RAM (RBC)
310	Flash : Protection violation
350	Write MDR ???
355	PCM_CODE does not set
392	Error on board RAM (RBC)
PLC : Register Errors	
400	PL_CNTRL_A
401	PL_CNTRL_B
402	PL_INTR_MASK
403	PL_XMIT_VECTOR
404	PL_VECTOR_LEN
405	PL_LE_THRESHOLD
406	PL_C_MIN
407	PL_TL_MIN
408	PL_TB_MIN
409	PL_T_OUT
410	PL_LC_LENGTH
411	PL_T_SCRUB
412	PL_NS_MAX
PLC : Other Errors	
420	Scrambler needed but not present
Special Hardware Dependent Errors	
450	Counter Test error
451	VPD error
452	Reset Test error
453	Watchdog error
550	bit_sign() PL_RCV_VECTOR
608	Compare error during DMA read
609	Compare error during DMA write
612	PCI : Timeout EOB not signaled
651	PCI : FORMAC parity check failure during send
652	PCI : FORMAC parity generation failure during receive
653	PCI : Parity status already set
654	PCI : PCI Parity error
655	PCI : mem_mapped test error
656	PCI : test_mode test error
657	PCI : BMU BIT test error
658	PCI : CFG space writes via IO space
659	PCI : Error in Bank 0 test
660	PCI : Error in CTRL register tests
661	PCI : DMA single test: cannot complete DMA
662	PCI : Error in Power Management Capability tests
663	PCI : diag_bmu, Descriptor is not given back
670	PCI : VPD transfer does not complete
671	PCI : VPD write error

Code	Description
672	PCI : Voltage sensor value out of the limits
673	PCI : Voltage sensor defective
674	PCI : Temperature sensor value out of the limits
675	PCI : Temperature sensor defective
676	PCI : I²C transfer does not complete
677	PCI : FPROM VPD ROM size does not match I²C EEPROM size
Check Timer 82C54	
801	Timer 82C54 does not decrement
802	Timer IRQ does not occur
803	IRQ is pending
804	Hardware Timer IRQ does not occur
805	Timer clock was not correct
810	(check token status)
899	PCI : STF or EOF not set in loopback test
900	PCI : MSVALID is not set in loopback test
901	PCI : Memory status receive abort in loopback test
902	PCI : Received frame not valid in loopback test
903	PCI : E-Indicator set in loopback test
904	PCI : FORMAC rx len unequal descriptor len in loopback test
905	PCI : length error in loopback test
906	PCI : Wrong FC found in loopback test
907	PCI : C-Indicator set in loopback test
908	PCI : Cannot get dummy RxD in loopback test
909	PCI : Stop Master test failed during loopback test
910	PCI : Token Counter does not increment in loopback test
998	Data header or body defective in loopback test
999	Illegal size in loopback test

7. Contact Details

7.1 Contact Person

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

7.2 Physical Address

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

7.3 Postal Address

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

7.4 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/

7.5 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).