Credit Net Switch GPU Diagnostics

Version 2.1

Gary Depelteau

‘depeltea@bnr.ca’

This Document

This document outlines the functional requirements, design description and man-machine interface required for GPU diagnostics of the MasterCard in the CreditNet ATM switch currently being developed by Harvard University and Bell-Northern Research Limited.

- Bell Northern Research
- Harvard University
-
-

Functional description (FN) 1

Design description (DD) 1

Functional description (FN)

WARNING: Although the basic overall contents of this document are still relevant, specific details are out of date. The HW for the non-maskable interrupt (NMI) source register for example still has some bugs (at the time of writing) and as a result the test for it usually fails If in doubt, please contact the author.

1.0 Issue/change control

Table 1: Revision History

Issue	Date	Reason	Change	Author
0.0	Oct. 15, 1993	Initial release for internal review	First Draft	Gary Depelteau
1.0	October 22, 1993	Release for external review	Update	Gary Depelteau
1.1	February 2, 1994	Update	Decided to run all tests before VxWorks is downloaded.	
1.2	June 17, 1194	Update	Update to reflect minor changes.	Gary Depelteau
2.1	May 19, 1995	Update	Added out of date warning	Gary Depelteau

2.0 Activity synopsis

The purpose of this activity is to provide the functional requirements and design of Generic Processor Unit (GPU) Diagnostics. GPU Diagnostics is responsible for performing both start-up and hardware verification diagnostics on the central processing unit of the Credit Net Switch common equipment cards. GPU Diagnostics tests the parts of the GPU to ensure that they are functioning properly and are capable of carrying out their role in the switch.

3.0 Activity description

3.1 Introduction

GPU start-up diagnostics are executed while the GPU is booting and before it downloads and begins running the VxWorks real-time operating system. These diagnostics are responsible for guaranteeing that the GPU is sane enough to download software into RAM and begin executing that software properly. GPU start-up diagnostics tests the following (See "Figure 1 GPU Block Diagram" on page 3.):

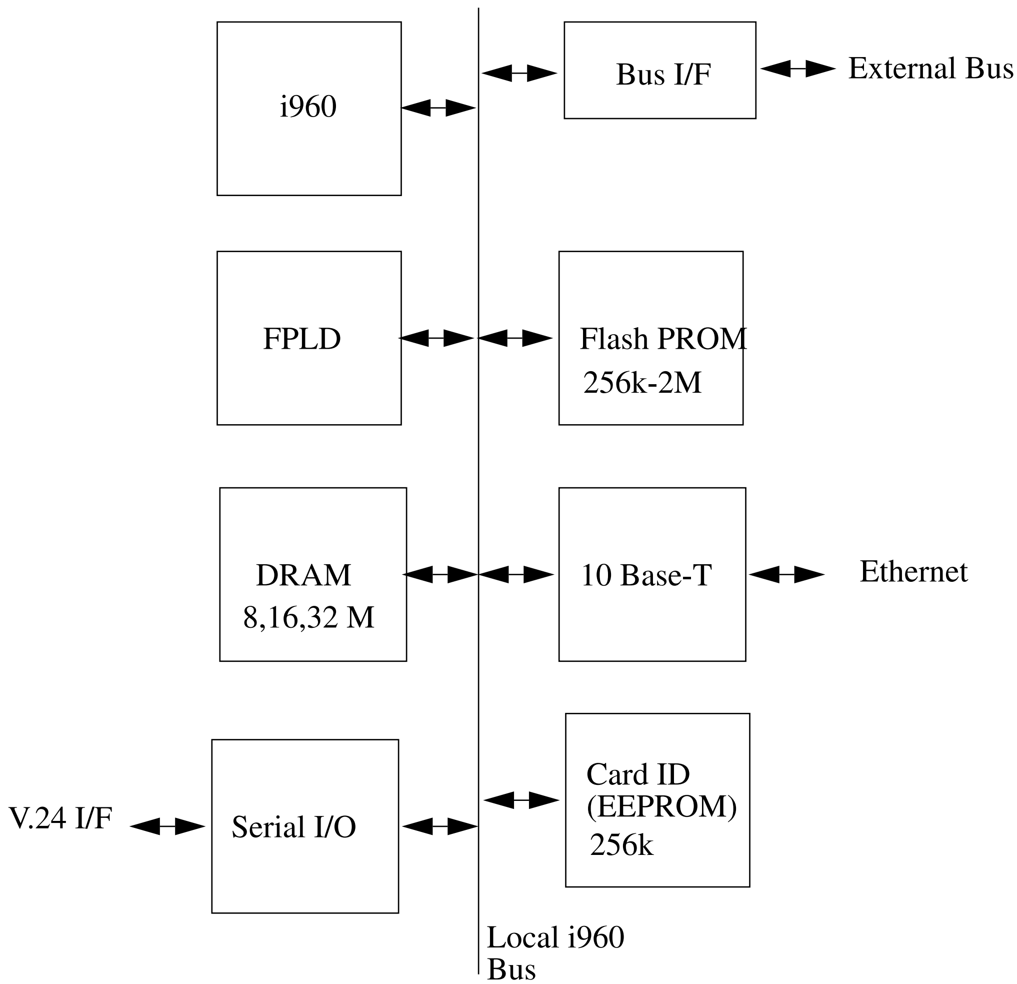
- Central Processing Unit (CPU) (an i960 built-in self test - no software development required);
- EEPROM;
- EEPROM Bank Swapping;
- Card Identification (Card ID);
- DRAM;
- Ethernet Controller;
- GPU Real-time Clock Interrupt.
- GPU Error/Status Registers;
- Serial Communication Controller (SCC).

GPU hardware verification diagnostics are executed at least once per board to guarantee that the boards were designed and manufactured correctly. All the resources tested by GPU start-up diagnostics are tested during hardware verification as well as the following GPU resources which can not be tested every time the switch boots:

- GPU External Bus;
- GPU DIP Switches.

Users can also execute any particular test individually from the VxWorks shell. This feature is very handy for debugging and fault isolation.

Figure 1 GPU Block Diagram



3.2 Functional specification - Start-up Diagnostics and Hardware Verification

From the VxWorks shell, it is possible to specify which blocks to test when invoking GPU Diagnostics by specifying a certain value for a test mask which is passed to the GPU Diagnostics controller. This controller is also responsible for determining the results of the tests performed while the GPU was booting (except for the EEPROM, Card Id and DRAM tests) and reporting any failures to the switch logging facility. It is not possible, however, to execute either of the DRAM tests via the GPU Diagnostics controller. The requirements for testing the GPU blocks defined earlier (See "Introduction" on page 1.) are covered in the following sections.

3.2.1 CPU - i960 Built-in Self Test

The i960 CA's Built-in Self Test is used to test the basic functionality of the processor bus, internal data paths, registers and memory arrays. Failure of this test causes the i960's FAIL pin to be asserted but no other registers are affected. Software has no means of intentionally detecting this failure with the present GPU design. This test is however useful in that the FAIL pin can be checked with a probe when hardware failures occur to see if the failure is caused by the i960. (If any software gets run by the i960 it is a good indication that the self test probably passed.)

3.2.2 EEPROM

The active bank of EEPROM is tested to verify that its contents, the bootstrap code, are correct. A CRC16 test is done on the EEPROM in order to do this.The EEPROM is also tested to guarantee that it contains the correct bank signature which indicates whether it is physically the primary or alternate bank.

3.2.3 Card ID (IDPROM)

The Card ID EEPROM contains both card identification information and records of serious errors. The Card ID is broken into many different blocks each with its own CRC16. GPU Diagnostics performs a CRC16 test on all the blocks of the Card ID.

3.2.4 DRAM

The DRAM test is required to test for stuck bits, leakage, address decoding errors and address and data line shorts. The DRAM is tested during bootstrap before data and software is downloaded into it. Two DRAM tests reside in EEPROM, a fast basic test and a longer more thorough test. The second more extensive test is able to pinpoint the address of DRAM errors and will only be run if the faster test fails or a DIP switch has been set indicating that this test should be run.

3.2.5 Ethernet Controller

The GPU's Intel 82596 Ethernet controller is tested in order to verify that the GPU is capable of communicating with the switch workstation via ethernet. This controller has a programmable built in self test which is performed in order to verify its sanity. It is necessary to verify that the results of this test, which are placed in DRAM, are correct and that the appropriate error registers do not indicate errors. Since the 82596's self-test results are placed in DRAM, it is necessary to test DRAM before performing this test.

(The 82586 also has a loopback function which can be used for diagnostics. This method of testing can be implemented if time permits or it becomes necessary.)

3.2.6 Timer Interrupt

The GPU's FPLD generates a 20 ms real time clock interrupt for the i960 in order that a real time operating system such as VxWorks can run. GPU Diagnostics verifies that this clock interrupt is occurring and at 20 ms intervals (give or take a small tolerance). The real time clock period is sampled five times in order to verify that it is the right length.

3.2.7 Error Registers

The GPU contains three error registers which are the Error Source Register, the Error Status Register, and the Error Address register. All of these register latch processor/board status when a non-maskable interrupt (NMI) occurs and require testing.

The Error Status register contains the following bits whose functionality must be verified:

1. General parity error bit;
2. Bus time-out bit;
3. Watchdog NMI bit;
4. Reset bit;
5. 0
6. 0
7. 0
8. NMI occurrence bit.

The ERROR Status Register contains the following bits:

1. BE0;
2. BE1;
3. BE2;
4. BE3;
5. R/W;
6. Ethernet Controller active;
7. DMA active;
8. External bus cycle.

The Error Address Register contains the 30 address bit settings that were present when an NMI occurred.
GPU Diagnostics causes events forcing the above bits to be set and then verifies that the corresponding bits have been set. Bits are then cleared after their tests have completed.

3.2.8 Control Status Register

The GPU also contains a Control Status register which reports the status of the following hardware control conditions:

1. 0
2. 0
3. 0
4. 0
5. Flash bank swapping jumper in place bit;
6. NMI disabled bit;
7. Flash programming enabled bit;

8. Flash software bank swapping enabled bit

GPU Diagnostics causes events forcing the above bits to be set and then verifies that the corresponding bits have been set. Bits are then to their original states after their tests have completed.

3.2.9 SCC

The GPU's AMD Z85C30 SCC is tested in order to verify that the GPU can communicate correctly with the console screen on the switch workstation via its serial port. The Z85C30 can be placed in loopback mode and this is used to verify that the Z85C30 can successfully send and receive a known character string. This test is performed while the Z85C30 interrupt is disabled so that the i960 is not interrupted.

3.2.10 Flash Bank Swapping

The GPU contains primary and backup secondary flash banks housing the VxWorks bootrom. It is possible via software control to swap between these two banks and this test verifies that that is possible.

3.3 Functional specification - Hardware Verification

Hardware verification test are invoked from the VxWorks Shell prompt and require human intervention. They are not performed every time the switch boots.

3.3.1 External Bus Test

This test requires that a special test jig, which loops the external bus address pins to the data pins, be connected to the external bus connectors. This test performs a series of read cycles from various external bus address (not decoded by the GPU) and verifies that the address bits are correctly looped back to the data bits.

3.3.2 Dip Switch Test

This test performs a walking 1 test on the GPU's eight DIP switches by prompting a user to turn the DIP switches on and off one at a time. This test has the ability to time out the responses from the user and report on the accuracy of the DIP switch values read.

3.4 Operations

3.4.1 .System initiated actions

Not applicable.

3.4.2 User initiated actions

During System Initialization, all tests outlined in the"Functional specification - Start-up Diagnostics and Hardware Verification" are executed. However, a human user working at the switch workstation has the ability to invoke the GPU Diagnostics controller from the VxWorks Shell (during switch development) in such a way that only specific GPU start-up diagnostics tests are executed. The user can also invoke any start-up or hardware verification tests individually from the VxWorks Shell.

3.5 Application interfaces

The entities invoking GPU Diagnostics are described in a previous section. (See "User initiated actions" on page 7.) GPU Diagnostics is invoked by calling the GPU Diagnostics controller passing it a test mask. The test mask contains one bit for each start-up test and determines which tests are executed.

GPU Diagnostics interfaces with the following software entities:

- The GPU-UI interface function, passing it a message number, to communicate results to users;
- The MARCH6 test function which is part of the System Initialization utilities module;
- The CRC16 test function which is also a part of the System Initialization utilities module;

3.6 Restrictions/limitations

Not applicable.

4.0 Optionality

Not applicable.

5.0 Activity impact

This activity impacts on the CreditNet Switch GPU hardware debugging since the tests developed are used to diagnose the hardware. Once switches are in service, it will verify GPU sanity during System Initialization and can be used to isolate faults when hardware problems are encountered.

5.1 Associated activities

This activity interacts with System Initialization in the sense that it is a part of System Initialization and is invoked by System Initialization software. It also impacts the User Interface software since the User Interface is responsible for converting some of the results of GPU Diagnostics into a user readable form.

6.0 Future enhancements

This activity may have to be expanded to test a SCSI controller which may be interfaced to an optional hard disk in the future. It may also be necessary to cut down on the time spent performing GPU Diagnostics.

7.0 Glossary of definitions & abbreviations

ATM	:Asynchronous Transfer Mode
BNR	:Bell Northern Research
DRAM	:Dynamic Random Access Memory
EEPROM	:Electrically Erasable-Programmable Read Only Memory
GPU	:Generic Processor Unit
RAM	:Random Access Memory
ROM	:Read Only Memory
TBD	:To Be Determined

8.0 References

[1] Wind River Systems, “VxWorks Programmers Guide.” Release 5.1. February, 1993.

[2] Wind River Systems, “VxWorks Reference Manual.” Release 5.1. March, 1993.

Internal Documents:

[3] Gary Depelteau, BNR, “System Initialization FN”, Sept. 1, 1993.

[4] Gary Depelteau, BNR, “System Initialization DD”, Sept. 16, 1993.

[5] Dave Milton, BNR. “CreditNet GPU Specification.” Version 1.0

Design description (DD)

WARNING: Although the basic overall contents of this document are still relevant, specific details are out of date. The HW for the non-maskable interrupt (NMI) source register for example still has some bugs (at the time of writing) and as a result the test for it usually fails If in doubt, please contact the author.

1.0 Issue/change control

Table 1

Issue	Date	Reason	Change	Author
0.0	November 11, 1993	Initial release for internal review	First Draft	Gary Depelteau
1.0	December 5, 1993	Release for external review	Review corrections	Gary Depelteau
2.0	February 2, 1994	Update	Decided to run all tests before VxWorks is downloaded.	Gary Depelteau
2.1	September 5, 1994, May 19, 1995	Update	Added ‘gpuCntrlRegTest’, ‘gpuFlSwapTest’, ‘gpuExtBusTest’, and ‘gpuDipSwTest’. Added out of date warning	Gary Depelteau

2.0 Design approach

2.1 Overview

GPU Diagnostics provides software for:

1. Generic Processor Unit (GPU) hardware verification;
2. GPU system start-up diagnostics;

2.2 Detailed Description

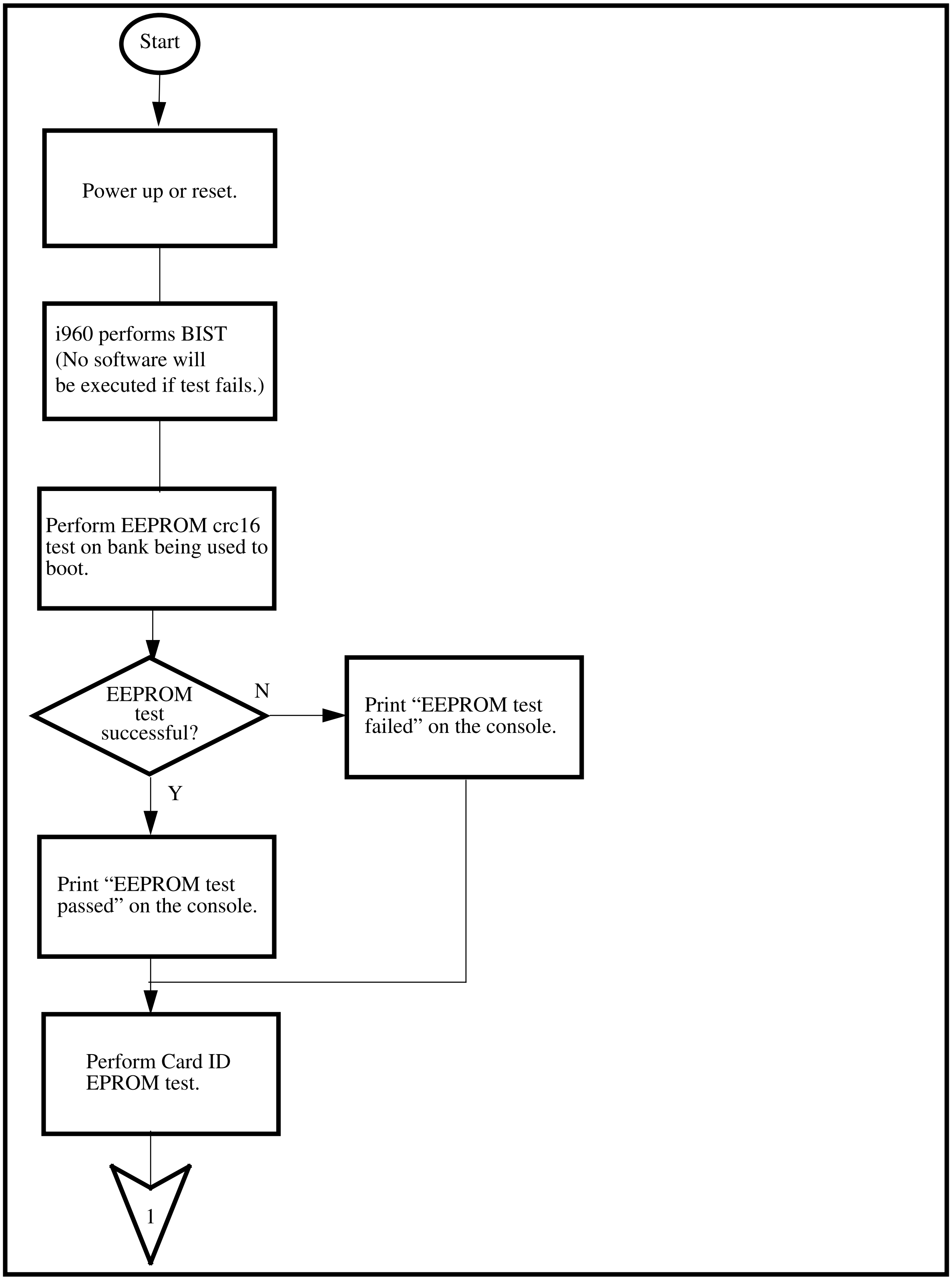
2.2.1 Bootstrap Phase Testing

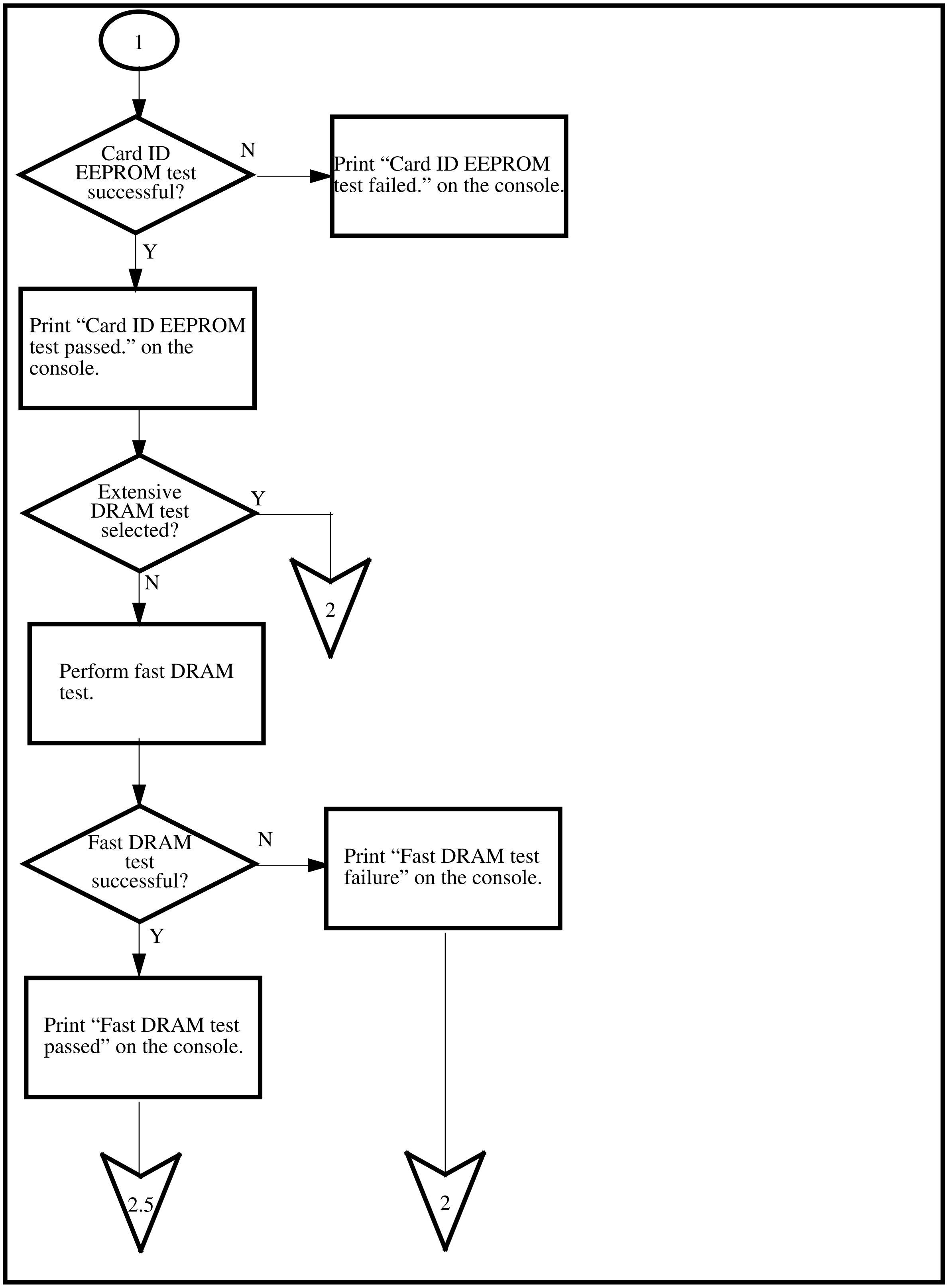
GPU start-up diagnostics are responsible for performing diagnostics on the central processing unit of the Credit Net Switch common equipment cards [5]. It tests the functional blocks of the GPU while running out of the bootrom (EEPROM) and before VxWorks and application software are downloaded to ensure that they are capable of performing their role in the switch. Messages indicating the outcome of each test are printed to the console on the switch workstation via the serial port. If the EEPROM, Card ID or DRAM tests fail, the processor will be placed in an endless loop once each of these tests has been tried. Users can force the GPU out of this endless loop by toggling one of the DIP switches. Users can also set another DIP switch in order to cause the extended DRAM test to be run instead of the fast DRAM test.

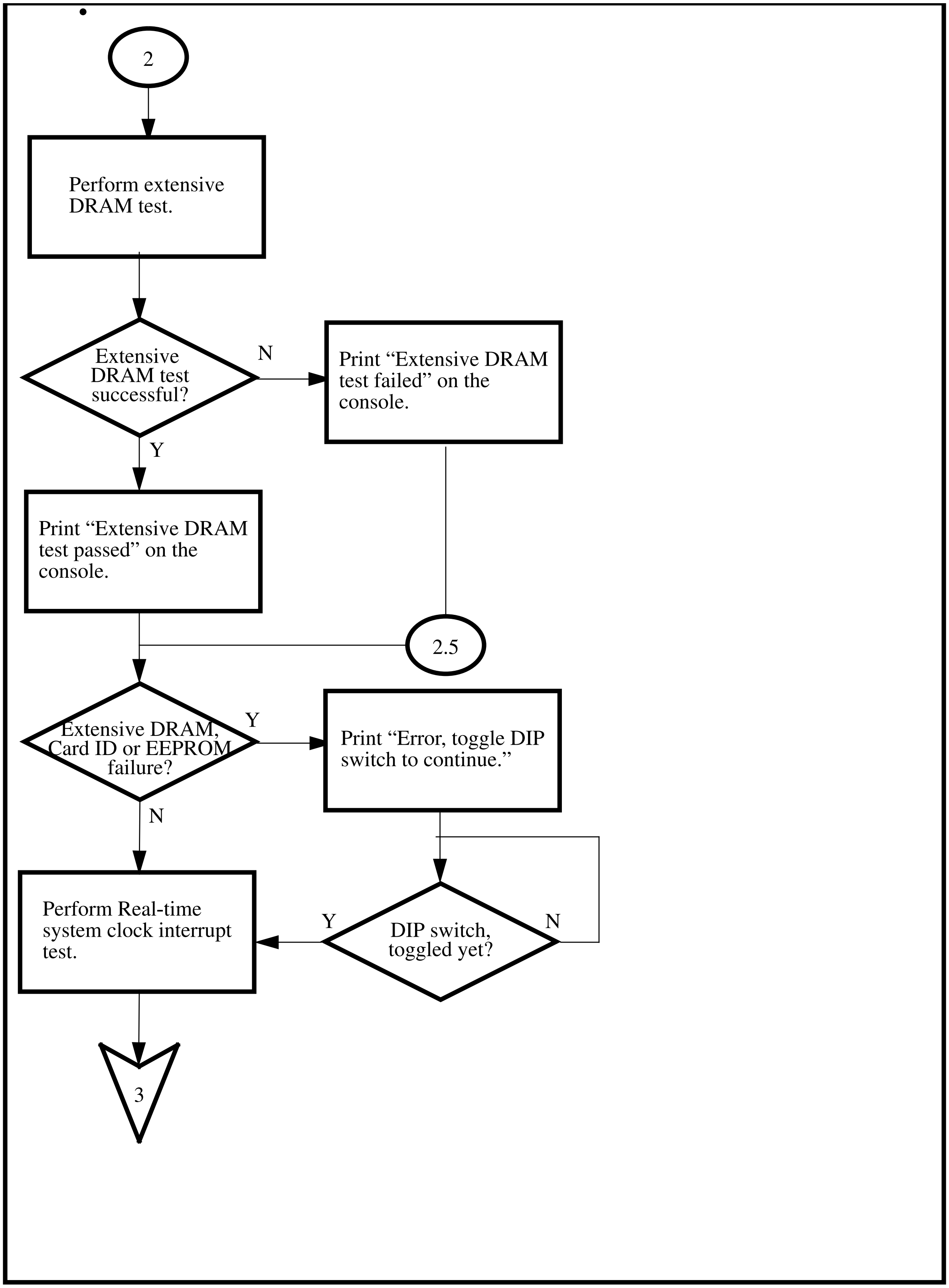
The following flowchart outlines the sequence of diagnostics events executed while the GPU is booting. For more information see "Figure 1 Bootstrap Phase Testing Flow Chart" on page 3.

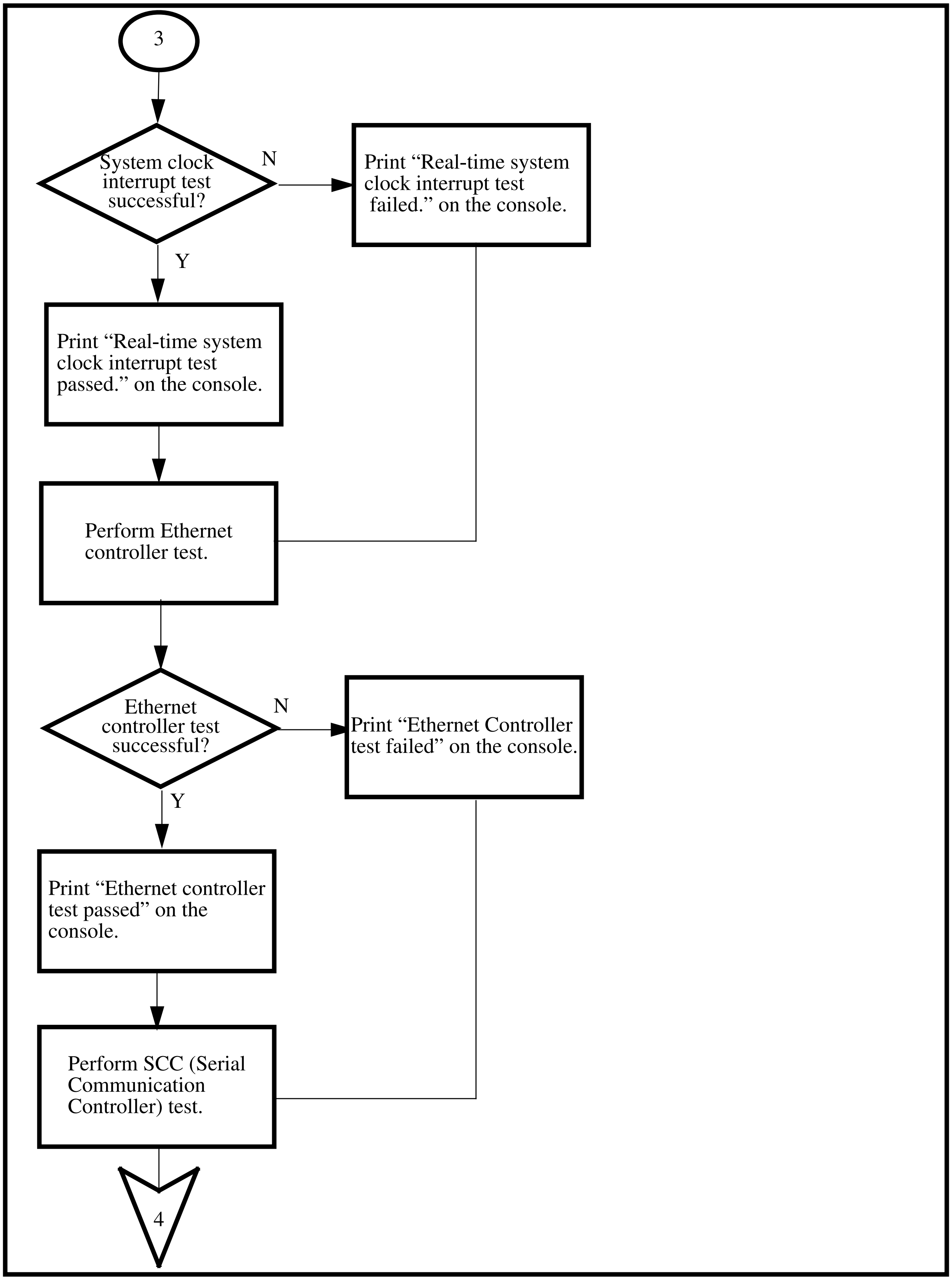
After System Initialization software has been downloaded, ‘gpuDiagCtl’ is invoked with a test mask instructing it to determine whether any of the tests performed during booting (except for EEPROM, Card ID and DRAM tests) failed. If any tests failed, this function then sends 32-bit error numbers to the switch User Interface (UI) informing it of the tests that failed.

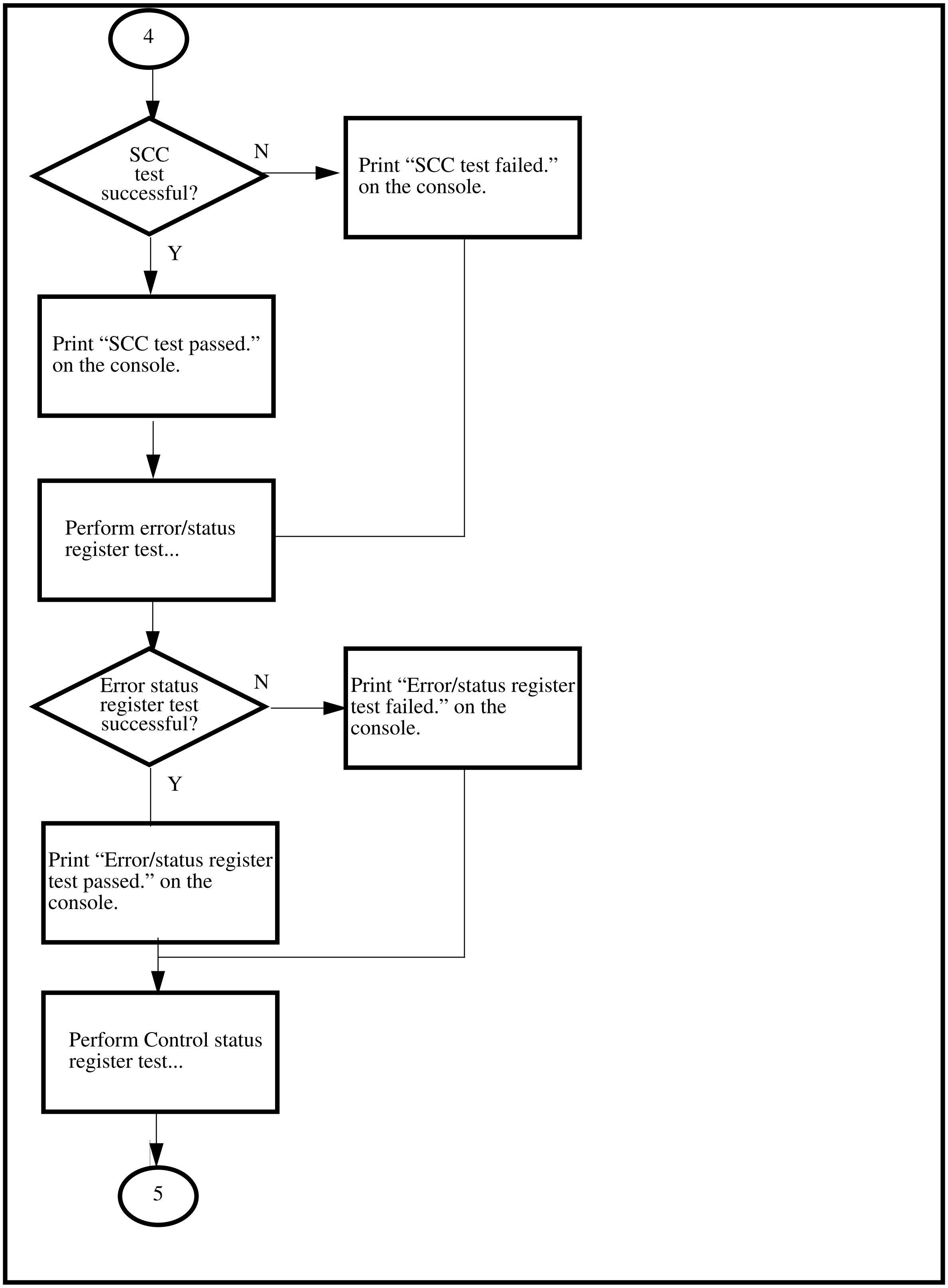
Figure 1 Bootstrap Phase Testing Flow Chart

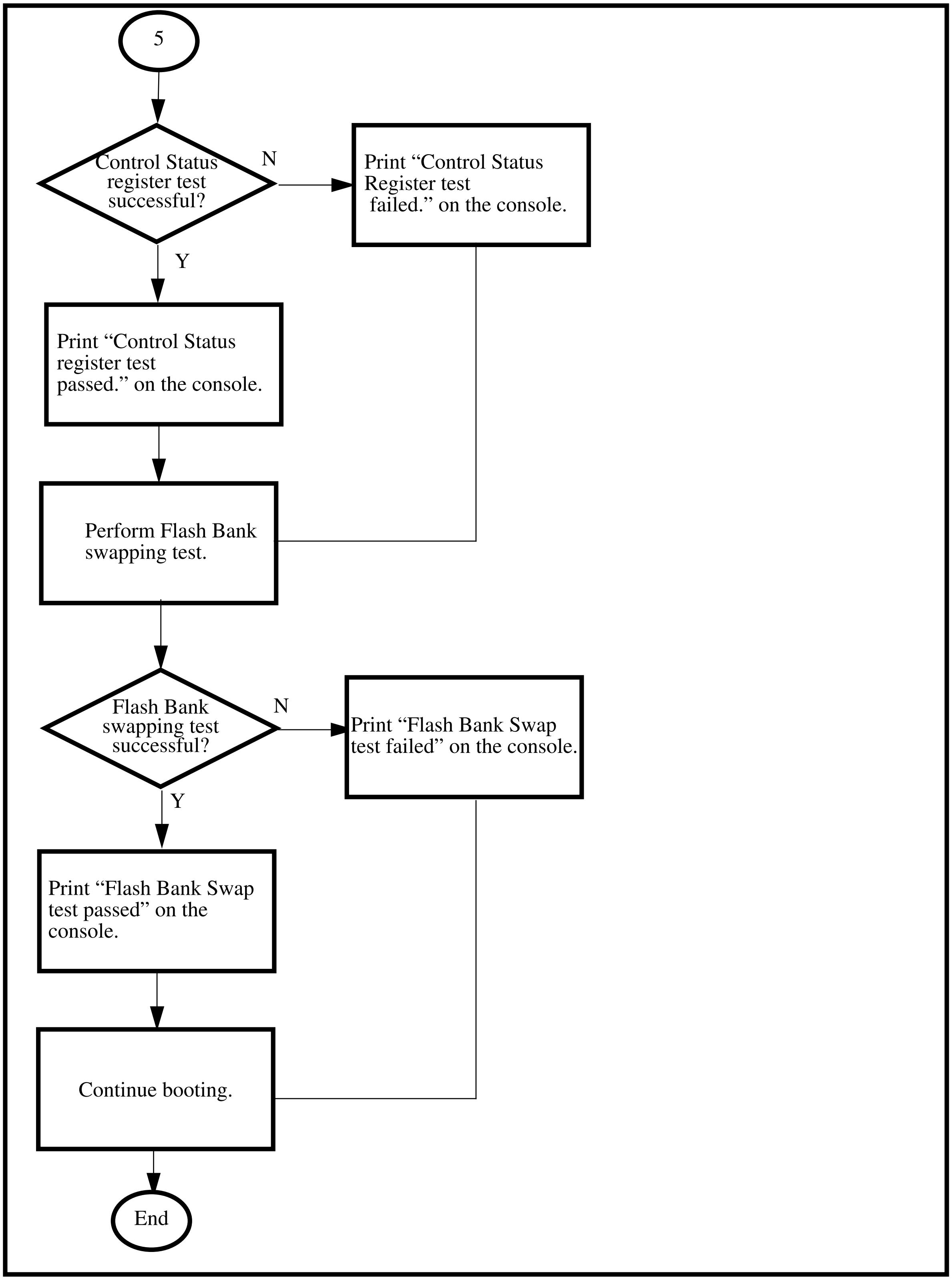



1
Card ID EEPROM test successful?
N
Print “Card ID EEPROM test failed.” on the console.
Y
Print “Card ID EEPROM test passed.” on the console.
Extensive DRAM test selected?
Y
2
N
Perform fast DRAM test.
Fast DRAM test successful?
N
Print “Fast DRAM test failure” on the console.
Y
Print “Fast DRAM test passed” on the console.
2.5
2


2
Perform extensive DRAM test.
Extensive DRAM test successful?
N
Print “Extensive DRAM test failed” on the console.
Y
Print “Extensive DRAM test passed” on the console.
2.5
Extensive DRAM, Card ID or EEPROM failure?
Y
Print “Error, toggle DIP switch to continue.”
N
Perform Real-time system clock interrupt test.
Y
DIP switch, toggled yet?
N
3


3
System clock interrupt test successful?
N
Print "Real-time system clock interrupt test failed." on the console.
Y
Print "Real-time system clock interrupt test passed." on the console.
Perform Ethernet controller test.
Ethernet controller test successful?
N
Print "Ethernet Controller test failed" on the console.
Y
Print "Ethernet controller test passed" on the console.
Perform SCC (Serial Communication Controller) test.
4


4
SCC test successful?
N
Print "SCC test failed." on the console.
Y
Print "SCC test passed." on the console.
Perform error/status register test...
Error status register test successful?
N
Print "Error/status register test failed." on the console.
Y
Print "Error/status register test passed." on the console.
Perform Control status register test...
5


5
Control Status register test successful?
N
Print “Control Status Register test failed.” on the console.
Y
Print “Control Status register test passed.” on the console.
Perform Flash Bank swapping test.
Flash Bank swapping test successful?
N
Print “Flash Bank Swap test failed” on the console.
Y
Print “Flash Bank Swap test passed” on the console.
Continue booting.
End

2.2.2 Hardware Verification Testing

Hardware verification testing is performed using both start-up GPU diagnostics test and specific hardware verification tests which require either special hardware setup of manual intervention. Any of the GPU Diagnostics can be invoked from the VxWorks Shell for the purpose of performing hardware verification testing. For more information see "External Interface Via VxWorks Shell" on page 29. If any tests failed, this function then sends 32-bit error numbers to the switch User Interface informing it of the tests that failed.

2.2.3 Initialization

Not Applicable.

2.2.4 Restart handling

Not applicable.

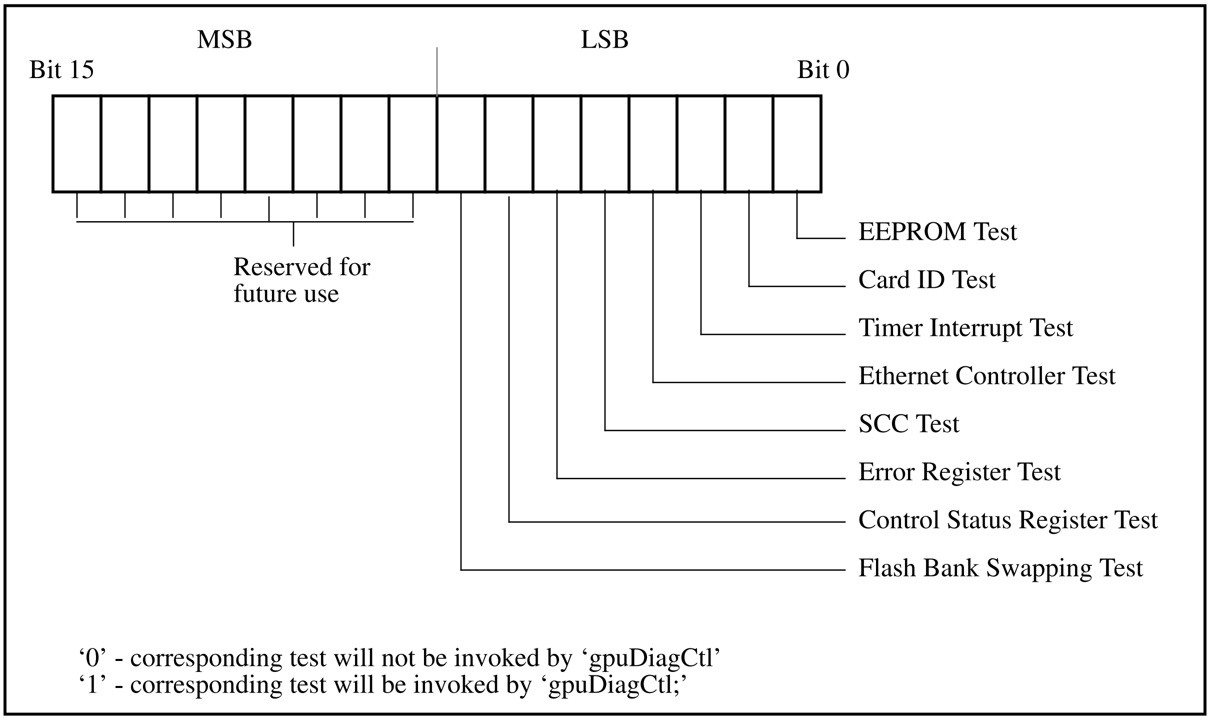
2.2.5 Data structures

2.2.5.1 GPU Test Mask

The GPU test mask is a short word (16 bits implemented as a short int in C) argument passed to function ‘gpuDiagCtl’ when it is invoked. It contains one bit for each GPU test that can be invoked (remember DRAM tests can not be invoked after the GPU boots) by ‘gpuDiagCtl.’.[1] ‘gpuDiagCtl’ will invoke all GPU tests whose corresponding bits are set in this mask. For more information see "Figure 3 GPU Test Mask" on page 10.

[1]Except for the DRAM tests which would not be safe to run after the GPU has downloaded software into DRAM.

Figure 3 GPU Test Mask

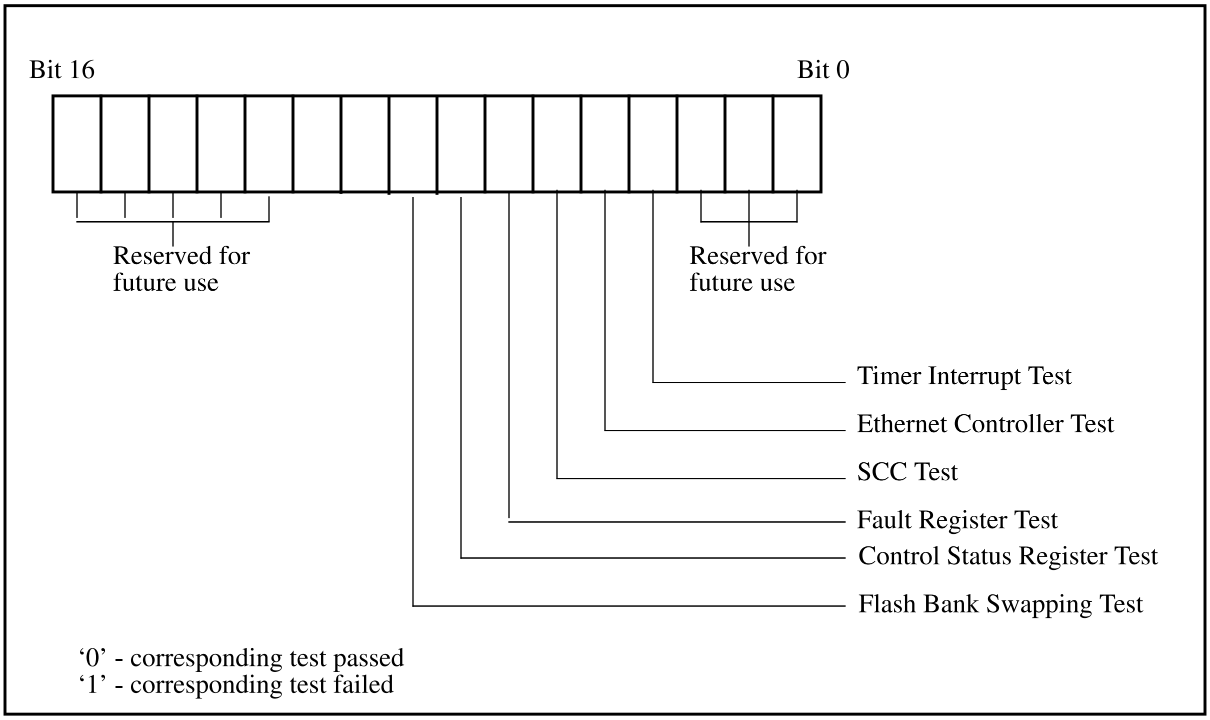


2.2.5.2 BOOT_RESULT_ADRS

BOOT_DIAG_RESULT_ADRS is at address 0x10000000 in DRAM and stores the results of the diagnostics run after the DRAM test is executed[2]. It is not practical to save the results of DRAM or earlier tests since this storage space may be corrupt if the DRAM test fails. It is this location that function 'gpuDiagCtl' uses to determine if any of the diagnostics run during booting failed so that it can send log numbers to the switch workstation indicating these errors. Individual bits are turned on in this memory location to indicate that certain tests failed.

[2]Address 0x10000000 to 0x10000FFF are not used by VxWorks.

Figure4 GPU960_BOOT_RESULT



2.3 Design evolution/strategy

Test functions will be developed and tested in the order in which they are executed during GPU booting. The control software residing in function 'gpuDiagCtl' will be implemented last.

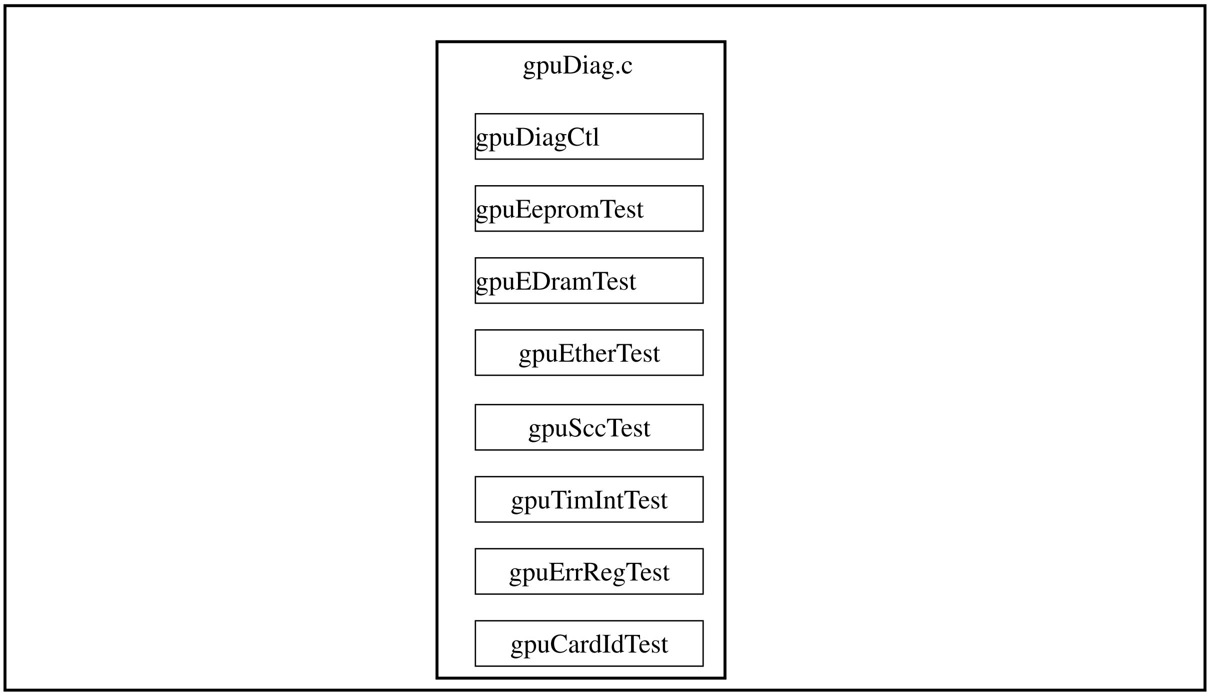
3.0 Software specification

3.1 Module 'gpuDiag.c'

3.1.1 Modular structure

All GPU Diagnostics software is contained in module 'gpuDiag.c' with the exception of the fast memory test 'fDramTest' which resides in the assembly module 'fDramTest.s.' Module 'gpuDiag.c' is compiled as 'gpuDiag_res.o' in order to be linked to the VxWorks bootrom and run during GPU booting. It is also compiled as 'gpuDiag.o' and downloaded with the rest of the System Initialization software after the GPU boots so that tests can be invoked from the VxWorks Shell. It is compiled in two different ways so that various degrees of feedback can be given to the user depending on whether the tests are being run while the GPU is booting or after that. It is also compiled in this way so that the test for GPU hardware verification are not included in 'gpuDiag_res.o'. This module is composed of function 'gpuDiagCtl' as well as individual functions implementing each GPU test. For more information see "Bootstrap Phase Testing" on page 2.

Figure 4 GPU Diagnostics Phase Testing Modular Structure



3.1.2 Function 'gpuDiagCtl'

3.1.2.1 Purpose

The purpose of 'gpuDiagCtl' (GPU Diagnostics Controller) is to determine the results of the diagnostics run while the GPU was booting and to provide a means of controlling the execution of diagnostics when they are invoked from the VxWorks Shell or other software.

3.1.2.2 Function

'gpuDiagCtl' performs the following functions:

1. Compares the values of the GPU Test Mask with acceptable values.[3] If it specifies an unimplemented test then:
    - Inform the user at the switch work station that the test mask specifies currently unimplemented test;[4]
    - Clear the bits in the test mask associated with unimplemented tests (those reserved for future use);

[3]This must be done since some bits in the GPU Test Mask are reserved for future use and may be accidently set by the user or calling function.

[4]Initially, during the development stage, this message should be passed to the user via the console where the VxWorks Shell is visible. In future phases of switch development, this message should be sent to the user via the (Graphical) User Interface.

    - Inform the user of the value of the test mask that is being used once bits associated with unimplemented test have been cleared;
2. If the GPU Test Mask is 0x0000 then:
    - Read the contents of BOOT_RESULT_ADRS;
    - For each bit set corresponding to the failure of a GPU diagnostics test, send the appropriate log number to the switch workstation using VxWorks function ‘logMsg’.[5]
3. For each bit set in the test mask:
    - Invokes the corresponding test;
    - If the test returns OK then print a message out to the VxWorks Shell indicating the test passed. If not, then print a message that the test failed.;
    - If the test returns ERROR then send a message number to the Switch UI by calling function ‘gpuUIErrorSend’ indicating that the corresponding test fails and set return status to ERROR. Also, if this function is being called while booting then set the corresponding bit at the BOOT_RESULT_ADRS;

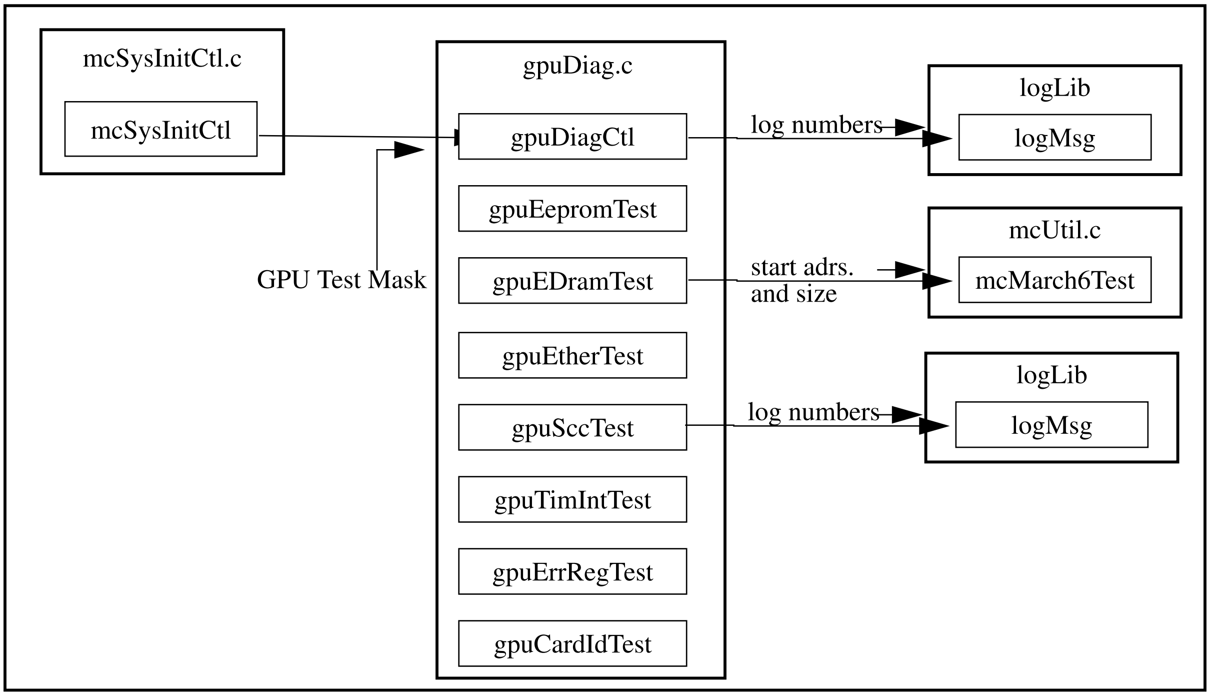
3.1.2.3 External interfaces

‘gpuDiagCtl’ is invoked externally by function ‘scmSysInitCtl’ which passes it the value of the GPU Test Mask. For more information see "Figure 3 GPU Test Mask" on page 10. It also invokes function ‘gpuUIErrorSend’ in ‘gpuUIinterface.c’. For more information see "Figure 3 GPU Test Mask" on page 10.

‘gpuDiagCtl’ may also be invoked from the VxWorks Shell. For more information see "External Interface Via VxWorks Shell" on page 29.

[5]‘logMsg’ queues a formatted string message for VxWorks function ‘logTask’ which sends it to a specified file descriptor. In this case, the file descriptor should specify a file on the switch workstation. [4]

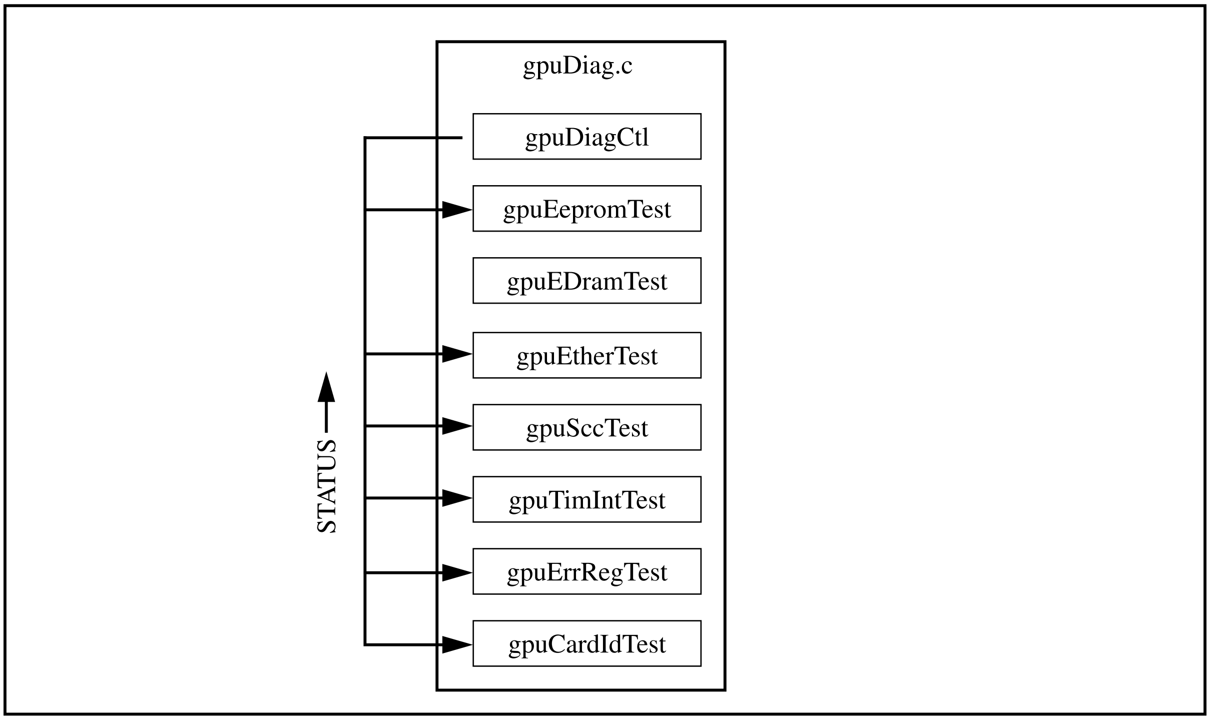
Figure 5 Module ‘gpuDiag.c’ External Interface



3.1.2.4 Internal interfaces

‘gpuDiagCtl’ internally invokes the GPU test functions. For more information see "Figure 3 GPU Test Mask" on page 10. They in turn return STATUS (OK or ERROR) when they complete.

Figure 6 Function ‘gpuDiagCtl’ Internal Interface



3.1.3 Function ‘gpuEepromTest’

3.1.3.1 Purpose

The purpose of ‘gpuEepromTest’ (GPU EEPROM Test) is to verify the reliability of the active EEPROM bank’s contents by performing a crc16 test.

3.1.3.2 Function

‘gpuEepromTest’ performs the following:

1. Calls function ‘crc16’ in ‘crc16.c’ to perform a crc16 calculation over the entire active EEPROM bank except for the last two bytes;
2. Compares the calculated crc16 to the expected value which is stored in the last two bytes of the EEPROM;
3. Returns OK if the calculated crc16 equals the expected value and returns ERROR otherwise.

3.1.3.3 External interfaces

‘gpuEepromTest’ may be invoked from the VxWorks Shell. It in turn calls function ‘crc16’ in ‘crc16.c.’ For more information see "External Interface Via VxWorks Shell" on page 29.

3.1.3.4 Internal interfaces
‘gpuEepromTest’ is internally invoked by function ‘gpuDiagCtl’. It in turn returns STATUS (OK or ERROR) when it completes. For more information see "Figure 3 GPU Test Mask" on page 10.

3.1.4 Function ‘gpuCardIdTest’

3.1.4.1 Purpose
The purpose of ‘gpuCardIdTest’ (GPU Card ID Test) is to verify the reliability of the contents of the Card ID EEPROM which is used to determine how the GPU and the it is a part of boots.

3.1.4.2 Function
The Card ID is filled with information blocks which start with a 16 bit size indicating the size of each block in bytes. The last two bytes of this each block contain a crc16 value for the block. ‘gpuEepromTest’ performs the following:

1. Initializes a block pointer to the beginning of the Card Id EEPROM;
2. reads the size of the block currently pointed to into blockSize;
3. Calls function ‘crc16’ in ‘crc16.c’ to perform a crc16 calculation over the blockSize -2 bytes pointed to by the block pointer. If the value returned by ‘crc16’ is not the same as the crc16 at the end of the block then return ERROR;
4. Increments the block pointer by the current blockSize;
5. If the new blockSize is 0xFFFF then return OK. Otherwise go to step 1.

3.1.4.3 External interfaces
‘gpuCardIdTest’ may be invoked from the VxWorks Shell. It in turn calls function ‘crc16’ in ‘crc16.c.’ For more information see "External Interface Via VxWorks Shell" on page 29.

3.1.4.4 Internal interfaces
‘gpuCardIdTest’ is internally invoked by function ‘gpuDiagCtl’. It inturn returns STATUS (OK or ERROR) when it completes. For more information see "Figure 3 GPU Test Mask" on page 10.

3.1.5 Function ‘gpuEDramTest’

3.1.5.1 Purpose
The purpose of ‘gpuEDramTest’ is to perform a more extensive test than ‘gpuFDramTest’. It provides more thorough coverage of address decoding as well as returning the address of the first memory location that is not functioning correctly.

3.1.5.2 Function

'gpuEDramTest' performs the following:

1. Calls function 'mcMarch6Test' in module 'mcUtil.c' with boolean check DRAM PE set to true and a foreground of 0xaaaaaaaa. 'mcMarch6Test' is a general purpose memory test which can be used to test various memories [6]. It for DRAM parity errors on each memory access since the boolean check DRAM PE is set to true. Passing 'mcMarch6Test' a foreground of 0xaaaaaaaa results in the parity bits in the DRAM being turned on for the entire test. 'gpuEDramTest also passes this function the start address of DRAM, its size and width (4 bytes). The March 6 test is composed of the following steps:

```
1. For base_address to end_address
    Write background (background=hFFFF XOR foreground);

2. For base_address to end_address
    Test the word for background;
    If (check DRAM PE) then check for parity error;
    Write foreground;

3. For base_address to end_address
    Test the word for foreground;
    If (check DRAM PE) then check for parity error;
    Write background;

4. For end_address to base_address (reverse address)
    Test the word for background;
    If (check DRAM PE) then check for parity error;
    Write foreground;

5. For end_address to base_address
    Test the word for foreground;
    If (check DRAM PE) then check for parity error;
    Write background;

6. For end_address to base_address
    Test the word for background;
  If (check DRAM PE) then check for parity error;
```

2. If 'mcMarch6Test' returns an address in the DRAM's address range then this indicates the first memory location at which an error was found. If this happens then 'gpuEDramTest' returns this address;
3. Fills the entire region with 0x00000001. This turns the DRAM parity bits off[6];
4. Checks the entire memory region to ensure that it contains 0x00000001. If any location does not contain 0x00000001 then it returns the address of this location. Also calls 'mcChkDramParity' in module 'mcUtil.c' and returns the address if a parity error is found;
5. Calls function 'mcMarch6Test' again with a foreground of 0xffffffff. This will cause the DRAM parity bits to be turned off;

[6]Steps 3 and 4 are necessary to ensure that the DRAM parity bits can change from 1 to 0 and 0 to 1. Both calls to 'mcMarch6Test' result in the DRAM parity bits being 1 throughout the test so these steps are needed to force the parity bits to be set to 0.

6. If ‘mcMarch6Test’ returns an address in the DRAM’s address range then this indicates the first memory location at which an error was found. If this happens then ‘gpuEDramTest’ returns this address;
7. Return OK

3.1.5.3 External interfaces

This function is called from function ‘romInit’ in module ‘romInit.s’ if the DIP switch bit selecting DRAM test functions is set for extensive DRAM testing. ‘romInit’ passes it the start address of DRAM and its size. It returns a status value upon return.For more information see "Figure 3 GPU Test Mask" on page 10. This function also calls ‘mcChkDramParity’ and receives STATUS (OK or ERROR) from it depending on whether or not a DRAM parity error was found.

3.1.5.4 Internal interfaces

Not applicable.

3.1.6 Function ‘gpuEtherTest’

3.1.6.1 Purpose

The purpose of ‘gpuEtherTest’ (GPU Ethernet Controller Test) is to verify the sanity of the GPU’s 82596CA LAN Coprocessor before it is used to load the VxWorks image and the rest of the switch software.

3.1.6.2 Function

‘gpuEtherTest’ sets a return status variable to ERROR at any point in the test when an error is detected, otherwise it remains set to OK. It performs the following:

1. Initializes return status to OK;
2. Masks NMI interrupt;
3. Writes to the NMI Error Clear register to clear the Error Source register[10];
4. Resets the 82596CA using a PORT command[2];
5. Assuming self-test results are to be written to 0x10400000 and 0x10400004 in DRAM while booting or from eight allocated bytes (via ‘malloc’) when run after booting. It fills these two locations with ones (ie. 0xffffffff);
6. Issues the self-test PORT command through the following steps:
    - Writes 0x0001 to the lower 2 bytes of the PORT access location;
    - Writes 0xC000 to the upper 2 bytes of the PORT access location;
7. Polls the self-test results addresses for a maximum of 100 ms until their contents change from their original values of 0xffffffff;

8. If no results are obtained in 100 ms or location or the second word of the self-test results is not 0x00000000 then it sends an error message to the console and sets return status to ERROR;
9. Checks the contents of the GPU Error Source register to see if the Ethernet parity error bit has been set. If it is set then this function sends an error message to the console and sets return status to ERROR;
10. Unmasks NMI interrupts;
11. Returns return status.

3.1.6.3 External interfaces

‘gpuEtherTest’ may be invoked from the VxWorks Shell. For more information see "External Interface Via VxWorks Shell" on page 29.

3.1.6.4 Internal interfaces

‘gpuEtherTest’ is internally invoked by function ‘gpuDiagCtl’. For more information see "Figure 3 GPU Test Mask" on page 10. It returns STATUS (OK or ERROR) when it completes.

3.1.7 Function ‘gpuSccTest’

3.1.7.1 Purpose

The purpose of ‘gpuSccTest’ (GPU SCC Test) is to verify the sanity of the AMD 85C30 Serial Communications Controller which is the controller portion of the serial port used to communicate with the switch workstation’s serial port. This serial channel will enable communication between the VxWorks Shell running on the GPU and the console window on the switch workstation. It is not tested during Bootstrap Phase diagnostics since it is not used during the downloading of VxWorks and switch software.

3.1.7.2 Function

‘gpuSccTest’ basically places the SCC in loopback, forces it to send a string to itself and checks it. It sets a return status variable to ERROR at any point in the test when an error is detected, otherwise it remains set to OK. It performs the following:

1. Initializes return status to OK;
2. Initializes an array of test characters to be used for loop-back;
3. Saves characters already in the SCC’s receive FIFO so that none will be lost[7];
4. Disables SCC interrupts so that other software can not interrupt the test and waits for any character that might have had a chance to slip in while interrupts are being disabled;
5. Waits 100 ms for the SCC to complete sending all characters in its transmit FIFO;

[7] This is necessary in case the SCC is not complete sending the last characters that it was instructed to transmit. Such a situation could arise if this test is invoked from the VxWorks Shell.

6. Turns on local loopback;
7. Transmits each character in the test character array back to the SCC via loopback;
8. Waits for a small amount of time for the test characters to be looped-back. If the test characters are not received in this small amount of time, it sets the return status to ERROR;
9. Checks the characters received and sets return status to ERROR if they are not the same as the original test characters;
10. Restores the characters that were originally in the receive FIFO by transmitting them back to the receive FIFO via loopback;
11. Turns off local loopback;
12. Re-enables SCC interrupts so that software can use it again;
13. Returns return status.

3.1.7.3 External interfaces

‘gpuSccTest’ may be invoked from the VxWorks Shell. For more information see "External Interface Via VxWorks Shell" on page 29.

3.1.7.4 Internal interfaces

‘gpuSccTest’ is internally invoked by function ‘gpuDiagCtl’. For more information see "Figure 3 GPU Test Mask" on page 10. It in turn returns STATUS (OK or ERROR) when it completes.

3.1.8 Function ‘gpuTimIntTest’

3.1.8.1 Purpose

The purpose of ‘mcgpuTimIntTest’ (GPU Timer Interrupt Test) is to verify that the GPU’s real time clock source can generate clock ticks at 20 ms intervals.

3.1.8.2 Function

‘mcgpuTimIntTest’ performs the following:

1. Sets return status toOK;
2. Disables task rescheduling by calling VxWorks function ‘taskLock’;
3. Save the original real time clock Interrupt Service Routine (ISR) and installs a test ISR. This test ISR will check the real-time clock bit and sets the global variable rtcBitSet to OK if this bit is set or ERROR if it is not set.
4. Initializes rtcBitSet to 0x1.
5. Enters a polling loop to poll rtcBitSet until it is set to OK or ERROR. It increments a counter each time the polling loop finds rtcBitSet equal to 0x1;

6. Repeats the above polling loop until rtcBitSet has been found OK six times or until it has been found equal to ERROR. Every time rtcBitSet equals OK it stores the value of the counter in an array, resets it to zero and repeats step 5. If rtcBitSet is equal to ERROR then this function sends a log number to the workstation indicating the occurrence of a rtc interrupt without the rtc bit being set, sets returns STATUS to ERROR and continues at step 8;
7. Checks the last five counter values stored in the array. (The first value will not be significant since it could have started counting anywhere in the clock tick period.) These should indicate that the clock period is 20 ms plus or minus twice the length of time it takes to execute the loop. If any of these values do not fall in the above range, this function sets return status to ERROR. The length of the loop will be calculated from the length of each assembly level instruction in the loop. This loop will then check that the last five count values fall in the range (20ms/loop_time +/- 2%);
8. Restores the original real time clock ISR;
9. Re-enables task rescheduling by calling VxWorks function ‘taskUnlock’;
10. Returns return status.

3.1.8.3 External interfaces

‘gpuTimIntTest’ may be invoked from the VxWorks Shell. For more information see "External Interface Via VxWorks Shell" on page 29.

3.1.8.4 Internal interfaces

‘gpuTimIntTest’ is internally invoked by function ‘gpuDiagCtl’. For more information see "Figure 3 GPU Test Mask" on page 10. It returns STATUS (OK or ERROR) when it completes.

3.1.9 Function ‘gpuErrRegTest’

3.1.9.1 Purpose

The purpose of function ‘gpuErrRegTest’ (GPU Fault Register Test) is to verify that the following bits in the fault register are functioning correctly:

- Watchdog NMI bit;
- DRAM parity error bit;
- Ethernet parity error bit;
- External bus parity error bit;
- External bus time-out bit.

The power up and Watchdog reset bits are not tested by this function but are instead tested the first time software writes to the fault register.A message will be printed on the switch workstation console indicating that one of these bits was set. This occurs early in the bootstrap sequence.

3.1.9.2 Function

This function simply forces the events associated with each of the fault register bits to happen and verifies that the correct bits are set. ‘gpuErrRegTest’ performs the following:

1. Sets return status to OK;
2. Disables task rescheduling by calling VxWorks function ‘taskLock’. This is necessary to guarantee that other tasks will not run while this tests Non-Maskable Interrupt (NMI) (ISR) is installed;
3. Install an ISR to handle the NMI interrupts associated with the bits in the fault register for the duration of this test. It should simply write the contents of the fault register to a known memory location so that the testing procedure can examine them. This ISR will also perform a dummy read of the NMI Error Clear register to clear the NMI Error Source register;
4. Reads the Error Clear register to clear the Error Source register before starting;
5. Checks the Watchdog NMI bit by performing the following steps:
    - Writes to the location 0xC0003000 to reset the Watchdog timer;
    - Executes a loop lasting a little more than 1 s;
    - Checks that only the Watchdog NMI bit is set. If not then it changes return status to ERROR;
    - Sends a log to the workstation indicating that the Watchdog NMI bit test either passed or failed;
    - Writes to the location 0xC0003000 to reset the Watchdog timer;
6. Checks the DRAM parity error bit by performing the following steps:
    - Sets the DRAM parity selector register for even parity on the next DRAM access;
    - Writes a value to a location in DRAM;
    - Performs a dummy read of the same location;
    - Checks that only the DRAM parity error bit has been set. If not then it sets return status to ERROR;
    - Sends a log to the workstation indicating that the DRAM parity error bit test either passed or failed;
7. Checks the ethernet controller parity error bit by performing the following steps:
    - Sets the DRAM parity selector register for even parity on the next DRAM access;
    - Writes a value to a location in DRAM;
    - Invokes ethernet driver functions so that they will format a pointer to the DRAM location with bad parity into a frame and instruct the ethernet controller to transmit this frame. (Care must be taken to ensure that the driver functions do not attempt to copy the DRAM word to another memory location before transmitting the frame since this would lead to a DRAM parity error and not an ethernet controller parity

error);

- Checks that only the ethernet parity error bit is set. If not then it changes return status to ERROR;
- Sends a log to the workstation indicating that the ethernet parity error bit test either passed or failed;

8. Testing of the external time-out bit external bus parity error bit will be implemented when the personality portion is available and something is connect to the external bus.
9. Re-installs the original NMI ISR;
10. Re-enables task rescheduling by calling VxWorks function ‘taskUnlock’;
11. Returns return status.

3.1.9.3 External interfaces

‘gpuErrRegTest’ may be invoked from the VxWorks Shell. For more information see "External Interface Via VxWorks Shell" on page 29.

3.1.9.4 Internal interfaces

‘gpuErrRegTest’ is internally invoked by function ‘gpuDiagCtl’. For more information see "Figure 3 GPU Test Mask" on page 10. It returns STATUS (OK or ERROR) when it completes.

3.1.10 Function ‘gpuCntrlRegTest’

3.1.10.1 Purpose

The purpose of function ‘gpuCntrlRegTest’ (GPU Control Status Register Test) is to verify that the following bits in the Control Status register are functioning correctly:

- NMI Disable bit;
- Flash Program Enable bit;
- Flash SW Bank Swap bit;

3.1.10.2 Function

This function simply forces the events associated with each of the Control Status register bits to happen and verifies that the correct bits are set. It performs the following:

1. Sets return status to OK;
2. Disables task rescheduling by calling VxWorks function ‘taskLock’. This is necessary to guarantee that other tasks will not run while this tests Non-Maskable Interrupt (NMI) (ISR) is installed;
3. Install an ISR to handle the NMI interrupts associated with the bits in the fault register for the duration of this test. It should simply write the contents of the fault register to a known

memory location so that the testing procedure can examine them. This ISR will also perform a dummy read of the NMI Error Clear register to clear the NMI Error Source register;

4. Saves the original value of Control Status register;
5. Checks the NMI disable bit by performing the following steps:
    - Calls function ‘sysNmiDisable’ in ‘sysLib.c’;
    - Verifies that the NMI Disable bit is set and sets result to ERROR if it is not;
    - Calls function ‘sysNmiEnable’ in ‘sysLib.c’;
    - Verifies that the NMI Disable bit is clear and sets result to ERROR if it is not;
    - Calls function ‘sysNmiDisable’ in ‘sysLib.c’;
    - Verifies that the NMI Disable bit is set and sets result to ERROR if it is not;
6. Checks the Flash Program Enable bit by performing the following steps:
    - Writes 0x0 to the Flash Program Enable register to enable Flash Programming;
    - Verifies that the Flash Program Enable bit is set and sets result to ERROR if it is not;
    - Writes 0x1 to the Flash Program Enable register to disable Flash Programming;
    - Verifies that the NMI Disable bit is clear and sets result to ERROR if it is not;
    - Writes 0x0 to the Flash Program Enable register to enable Flash Programming;
    - Verifies that the NMI Disable bit is set and sets result to ERROR if it is not;
7. Checks the Flash Bank Swap Enable bit by performing the following steps:
    - Calls function ‘sysSoftFlSwap’ in ‘sysLib.c’ if SW bank swapping is already active;
    - Verifies that the Flash Bank Swap Enable bit is clear and sets result to ERROR if it is not;
    - Calls function ‘sysSoftFlSwap’ in ‘sysLib.c’;
    - Verifies that the Flash Bank Swap Enable bit set and sets result to ERROR if it is not;
    - Calls function ‘sysSoftFlSwap’ in ‘sysLib.c’;
    - Verifies that the Flash Bank Swap Enable bit is clear and sets result to ERROR if it is not;
8. Restores the original state of the Control Status Register;
9. Re-installs the original NMI ISR;
10. Re-enables task rescheduling by calling VxWorks function ‘taskUnlock’;
11. Returns return status.

3.1.10.3 External interfaces

‘gpuCntrlRegTest’ may be invoked from the VxWorks Shell. For more information see "External Interface Via VxWorks Shell" on page 29.

3.1.10.4 Internal interfaces

'gpuCntrlRegTest' is internally invoked by function 'gpuDiagCtl'. For more information see "Figure 3 GPU Test Mask" on page 10. It returns STATUS (OK or ERROR) when it completes.

3.1.11 Function 'gpuFlSwapTest'

3.1.11.1 Purpose

The purpose of function 'gpuFlSwapTest' (GPU Flash Bank Swapping Test) is to verify that the GPU can swap back an forth between the primary and secondary flash banks through the use of the software bank swapping enable register;

3.1.11.2 Function

'gpuFlSwapTest' performs the following:

1. Sets return status to OK;
2. Saves the original value of Control Status register;
3. Saves the bank signature of the current primary bank which is found at address 0xFFFFFFFE;
4. Calls function 'sysSoftFlSwap' in 'sysLib.c';
5. Saves the bank signature of the current primary bank which is found at address 0xFFFFFFFE;
6. Calls function 'sysSoftFlSwap' in 'sysLib.c';
7. Compares the two bank signatures just save and sets result to ERROR if they are the same;
8. Compares the present state of the Control Status Register to the current one and sets result to ERROR if they are not the same since this would mean original state of bank swapping could not be restored;
9. Returns return status.

3.1.11.3 External interfaces

'gpuFlSwapTest' may be invoked from the VxWorks Shell. For more information see "External Interface Via VxWorks Shell" on page 29.

3.1.11.4 Internal interfaces

'gpuFlSwapTest' is internally invoked by function 'gpuDiagCtl'. For more information see "Figure 3 GPU Test Mask" on page 10. It returns STATUS (OK or ERROR) when it completes.

3.1.12 Function ‘gpuExtBusTest’

3.1.12.1 Purpose

The purpose of function ‘gpuExtBusTest’ (GPU External Bus Test) is to verify the integrity of the GPU External Bus address and data bits; This function is only used during hardware verification testing and requires that a special test jig, which loops the address and data lines of the external bus together, be connected to the external bus;

3.1.12.2 Function

‘gpuExtBusTest’ performs the following:

1. Repeats the following:
    - Walks a one across the address bus from bits 2 to 27;
    - Reads from the current address;
    - Prints out the current address and the value read from it;

3.1.12.3 External interfaces

‘gpuExtBusTest’ may be invoked from the VxWorks Shell. For more information see "External Interface Via VxWorks Shell" on page 29.

3.1.12.4 Internal interfaces

‘gpuExtBusTest’ is internally invoked by function ‘gpuDiagCtl’. For more information see "Figure 3 GPU Test Mask" on page 10. It returns STATUS (OK or ERROR) when it completes.

3.1.13 Function ‘gpuDipSwTest’

3.1.13.1 Purpose

The purpose of function ‘gpuDipSwTest’ (GPU DIP Switch Test) is to verify the integrity of the GPU’s 8-bit DIP switch

3.1.13.2 Function

‘gpuDipSwTest’ performs the following:

1. Initializes result to OK;
2. Prompts the User to set all DIP switches to the on position;
3. Waits for the User to turn on all DIP switches. The function will time this out and return ERROR if all DIP switches are not turned on within a reasonable amount of time;
4. Prompts the User to turn DIP switch 1 off;
5. Waits for the user to turn DIP switch 1 off;

6. Prints OK if the reading the DIP switch register indicates that DIP switch 1 off or returns ERROR if this times out;
7. Prompts the User to turn DIP switch 1 on;
8. Waits for the user to turn DIP switch 1 on;
9. Prints OK if the reading the DIP switch register indicates that DIP switch 1 on or returns ERROR if this times out;
10. Repeats steps 4. through 9. for DIP switches 2 through 8;
11. Returns result;

3.1.13.3 External interfaces
'gpuDipSwTest' may be invoked from the VxWorks Shell. For more information see "External Interface Via VxWorks Shell" on page 29.

3.1.13.4 Internal interfaces
'gpuDipSwTest' is internally invoked by function 'gpuDiagCtl'. For more information see "Figure 3 GPU Test Mask" on page 10. It returns STATUS (OK or ERROR) when it completes.

3.2 Modular structure - 'fDramTest.s'
Module 'fDramTest.s' only contains function 'fDramTest' which was written in assembly for speed.

3.2.1 Function 'gpuFDramTest'

3.2.1.1 Purpose
The purpose of 'gpuFDramTest' (GPU Fast DRAM Test) is to provide a fast DRAM test that catches a large proportion of DRAM faults. It is the primary DRAM test executed during System Initialization when the GPU is booting. It is called from function 'romInit' before EEPROM contents are copied into DRAM 'gpuFDramTest' tests for the following:

1. Stuck memory cell bits;
2. Memory cell leakage;
3. Address-address line shorts;
4. Data-data line shorts;
5. Parity errors and stuck parity bits.

3.2.1.2 Function

'gpuFDramTest' performs the following:

1. Loops through entire DRAM filling it with 0xA5A5A5A5. This will turn the parity bits on;
2. Loops through entire DRAM checking each location for 0xA5A5A5A5;
3. Loops through entire DRAM filling each location with 0x5A5A5A5A while parity generation is set for even parity on each memory access.This will turn the parity bits off;
4. Loops through entire DRAM checking each location for 0x5A5A5A5 while parity generation is set for even parity on each memory access;
5. Loops through entire DRAM filling each location with 0x00000000. This will turn parity bits back on;
6. Checks GPU Error Status register for DRAM parity error and clears the register if found;
7. Loops through the 24 word locations of DRAM addressed by the pattern 0x10000000, 0x10000004, 0x10000008, 0x10000010, 0x10000020... 0x10100000 and fills them with their address values;
8. Check that the above patterns written in 6 are correct;
9. Write a walking one pattern to 32 consecutive addresses (ie. 0x00000001, 0x10000002...0x1000001F;
10. Check that the above patterns written in 8. are correct;
11. Checks GPU Error Status register for DRAM parity error and clears the register if found
12. Return OK.

This function returns ERROR as soon as any of the memory checks fail to yield the expected value or a parity error is discovered, otherwise it returns OK. This test is coded in i960 assembly language to guarantee that it uses only registers to temporarily store variables as well as to make it as fast as possible.

3.2.1.3 External interfaces

This function is called from function 'romInit' in module 'romInit.s' if the DIP switch bit selecting DRAM test functions is set for fast DRAM testing. 'romInit' passes it the start address of DRAM and its size. It returns a status value upon return. For more information see "Figure 3 GPU Test Mask" on page 10.

3.2.1.4 Internal interfaces

Not applicable.

3.3 External Interface Via VxWorks Shell

All functions in module ‘gpuDiag.c’ can be invoked by a user from the command line interface of the VxWorks Shell on the console of the switch workstation. When invoked in this manner, functions execute at the same priority level as the shell. These tests may also be spawned as separate tasks from the shell, using VxWorks function ‘taskSpawn’, and can be given a higher or lower priority than the shell. For more information on invoking functions from the shell, please see the VxWorks Programmer’s Guide.[8]

The ability to invoke functions from the shell will give the user the ability to perform any test in module ‘gpuDiag.c’ individually by calling the corresponding function or to execute any set of tests by calling function ‘gpuDiagCtl’ and passing it an appropriate test mask.

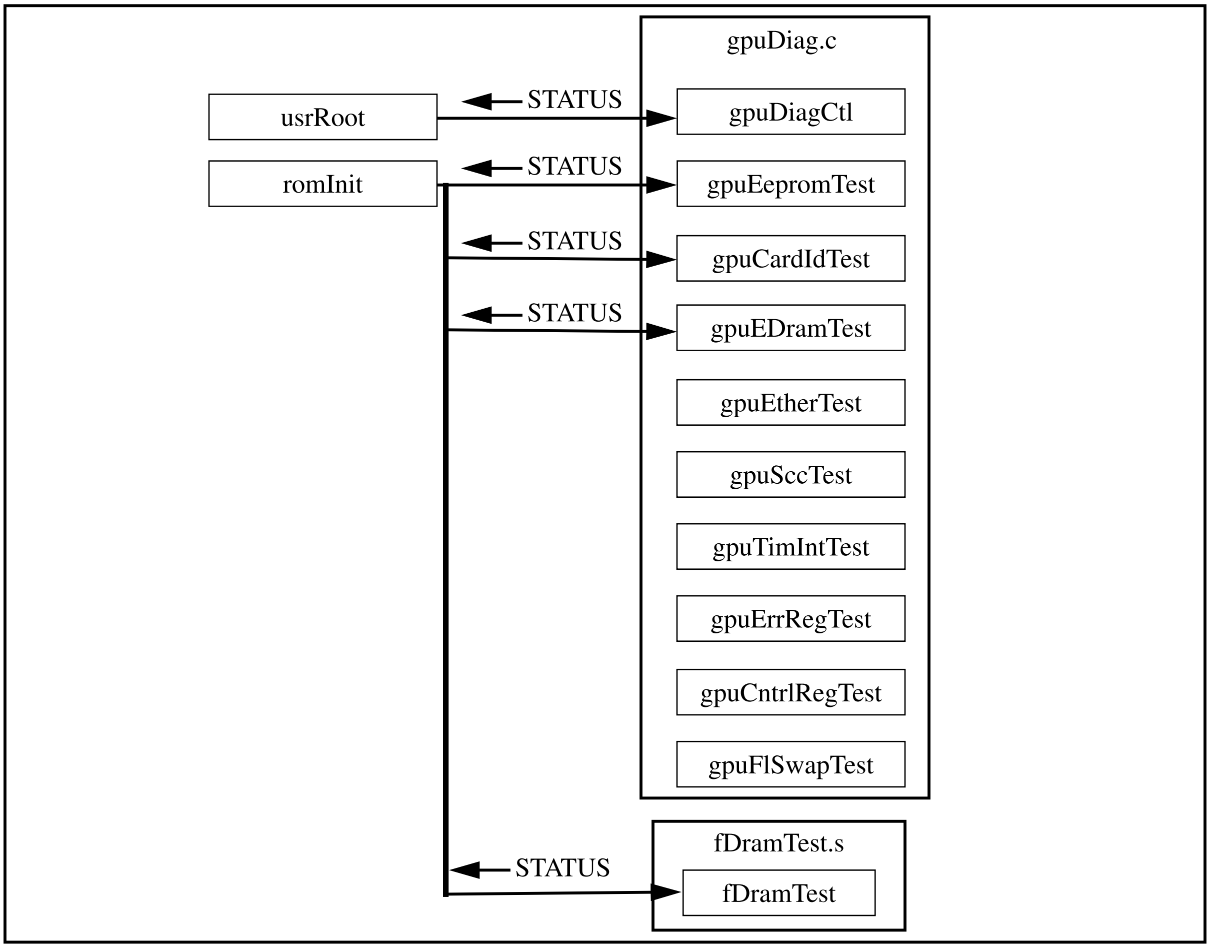
CAUTION: performing certain hardware diagnostics can have adverse affects on the GPU. Users should ensure that tests performed do not have ill effects on the current operation of the switch.

3.4 Modular structure - Bootstrap Phase Testing

Bootstrap phase testing is executed out of the bootrom (linked as ‘gpuDiag_res.o’) image so that the tests can be executed before any code has to be downloaded to the GPU. The fast memory test though, ‘fDramTest’ resides in module ‘fDramTest.s.’ These tests are invoked from ‘romInit’ and ‘usrRoot’ in ‘bootConfig.c’ which are part of the GPU’s VxWorks Board Support Package (BSP). For more information see "Figure 3 GPU Test Mask" on page 10.

[8]When switch development is complete, the VxWorks Shell should be removed and this functionality should be supplied by a (Graphical) User Interface.

Figure 7 Bootstrap Phase Testing - External Interface



3.4.1 Changes to ‘romInit’

3.4.1.1 Purpose

Function ‘romInit’ resides in VxWorks module ‘romInit.s’ and is the first function executed out of EEPROM when the processor is powered-up or reset.

3.4.1.2 Function

Changes to ‘romInit’ will provide the following functionality:

1. Initialize the contents of a register not used by either memory test to OK. This register will be used to hold the status of functions ‘gpuEepromTest’, ‘gpuCardIdTest’, and the memory test that gets run;
2. Calls function ‘gpuEepromTest’;
3. Writes ERROR into the contents of status register initialized in 1. if ‘gpuEepromTest’ returns ERROR and print a message to the console stating that this test failed. Otherwise print a message on the console stating that this test passed;

4. Calls function ‘gpuCardIdTest’;
5. Writes ERROR into the contents of status register initialized in 1. if ‘gpuCardIdTest’ returns ERROR and print a message to the console stating that this test failed. Otherwise print a message on the console stating that this test passed;
6. Calls either function ‘gpuFDramTest’ or ‘gpuEDramTest’ depending on the setting of a DIP switch on the GPU which determines the DRAM test to be executed. It passes the selected DRAM test the starting address of DRAM and its size;
7. If ‘gpuFDramTest’ was invoked and failed then call function ‘gpuEDramTest’ and print a message on the console stating that the fast DRAM test failed. If this test passed then print a message on the console stating that this test passed;
8. If ‘gpuFDramTest’ was invoked and failed print a message on the console stating that the fast DRAM test failed and write ERROR in the status register. If this test passed then print a message on the console stating that this test passed;
9. Examine the contents of the status register and continue booting if it contains OK. If the status register contains ERROR the prompt the user to toggle one of the DIP switches and sit in an idle loop until the switch is toggled.

3.4.1.3 External Interface

For more information see "Figure 3 GPU Test Mask" on page 10.

3.4.2 Changes to ‘usrRoot’ In Module ‘bootConfig.c’

3.4.2.1 Purpose

Function ‘usrRoot’ resides in VxWorks module ‘bootConfig.c’ and is responsible for initiating real-time clock, SCC and Ethernet controller initialization.

3.4.2.2 Function

Changes to ‘usrRoot’ in module ‘bootConfig.c’ will provide the following functionality:

1. Initializes the contents of BOOT_RESULT_ADRS to OK. This is actually the first location in DRAM and is used to store the results of the tests invoked in the following steps;
2. Calls function ‘gpuDiagCtl’ passing it a test mask indicating that it should perform start-up diagnostics;
3. If ‘gpuDiagCtl’ returns ERROR then this function sends a message to the Switch UI indicating that GPU Start-up Diagnostics failed. If not, then this function sends a message to the Switch UI indicating that GPU Start-up Diagnostics passed;

3.4.2.3 External Interface

For more information see "Figure 3 GPU Test Mask" on page 10.

4.0 Real time estimates

5.0 Glossary of definitions & abbreviations

ATM :Asynchronous Transfer Mode

BNR :Bell Northern Research

CDM :Cell Departure Monitor

CIF :Cell Interface

CMM :Core Memory Manager

CUM :Core Utilization Monitor

DRAM :Dynamic Random Access Memory

DP :Dual Port

EEPROM :Electrically Erasable-Programmable Read Only Memory

FIFO :First-In First-Out

FLM :Free List Manager

GPU :General Processor Unit

IVC :Internal Virtual Connection

LUT :Look-Up Table

NIS_INIT :Not In Service - Initialization State

NIS_NETWK :Not In Service - Network State

PBIF :Processor Control Bus Interface

PC :Port Card

PCC :Port Card Controller

R_IVC :Read IVC (lines on the switch backplane)

RAM :Random Access Memory

RCC :Round trip Cell Count

ROM :Read Only Memory

RTT :Round Trip Time

SP :Single Port

TBD :To Be Determined

UI :User Interface

VC :Virtual Connection

W_CADR :Write Cell Address (lines on the switch backplane)

W_IVC :Write IVC (lines on the switch backplane)

6.0 References

[1] Intel, “i960CA Microprocessor Users Manual.” 1992.

[2] Intel, “32-Bit LAN Component User’s Manual.” 1992.

[3] Wind River Systems, “VxWorks Programmers Guide.” Release 5.1. February, 1993.

[4] Wind River Systems, “VxWorks Reference Manual.” Release 5.1. March, 1993.

Internal Documents:

[5] Gary Depelteau, BNR, “System Initialization FN”, Sept. 1, 1993.

[6] Gary Depelteau, BNR, “System Initialization DD”, Sept. 16, 1993.

{7} Gary Depelteau, BNR, “GPU Diagnostics FN”, Sept. 16, 1993.

{8} Ken Grimble, BNR. “MasterCard Specification.” July 6, 1993.

[9] Gajendra Gupta, BNR, “OMEGA Generic Digital Diagnostics_A34”, May 17, 1991.

[10] Dave Milton, BNR. “CreditNet GPU Specification.” Version 1.0

[11] Margo Seltzer and Mike Smith, Harvard University.”Software Specification.” Version 0-2. April 13, 1993.

[12] Cliff Young, Harvard University. “Port Controller Specification.” Version 1-1. June 16,1993.